

# DS3500

---

Advanced Programming with Data



Northeastern University

John Rachlin  
Northeastern University

# Principles of Chess

## The Top 35 Chess Principles



<b>Control the Center</b>	<b>Doubled rooks on an open file are very strong</b>
<b>Develop pieces quickly</b>	<b>Bishops are better in open positions, knights are better in closed positions</b>
<b>Knights before bishops</b>	<b>The best way to deal with a flank attack, is with a counter attack in the center</b>
<b>Don't move the same piece twice in the opening</b>	<b>When 2 pawns can capture the same piece, capture towards the center</b>
<b>Don't bring your queen out too early</b>	<b>The king should be activated in the endgame</b>
<b>Castle before move 10</b>	<b>Rooks go behind passed pawns</b>
<b>Connect your rooks</b>	<b>2 connected passed pawns on the 6th rank will beat a rook</b>
<b>Rooks should go to open or half open files</b>	<b>Attack the base of a pawn chain</b>
<b>Knights on the rim are grim</b>	<b>Knights are usually the best piece to use to blockade a pawn</b>
<b>Try to avoid doubled pawns</b>	<b>If your position is cramped, trading pieces can help</b>
<b>Try to avoid isolated pawns</b>	<b>Trade your passive pieces for your opponent's active pieces</b>
<b>Try to avoid backward pawns</b>	<b>When ahead material, trade pieces, not pawns</b>
<b>Don't trade a bishop for a knight without a good reason</b>	<b>When behind material, trade pawns, not pieces</b>
<b>Avoid moving pawns in front of your castled king</b>	<b>Games with opposite colored bishops are dangerous in the middlegame, and drawish in the endgame</b>
<b>Don't open the center if your king is still there</b>	<b>Don't play "hope" chess</b>
<b>2 minor pieces are usually better than a rook and a pawn</b>	<b>When you see a good move, look for a better move</b>
<b>3 minor pieces are usually better than a queen</b>	<b>A really good chess player knows the right time to ignore a chess principle</b>
<b>Rooks are good on the 7th rank</b>	



Northeastern University

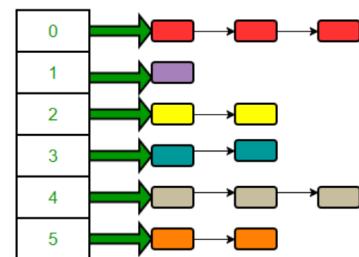
# DS 2000: Programming with Data

- DS 2000 is the first course in Northeastern's *Programming with Data* sequence.
- DS 2000 is for DS majors, minors, and non-CS / non-DS majors. *No programming experience is assumed!*
- *DS 2000 is the one course to take if you are only going to take one programming course, or you want a broad introduction to Data Science.*

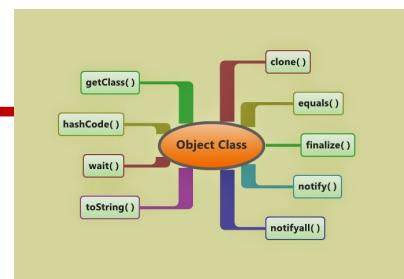
## Some of the topics explored in DS2000:



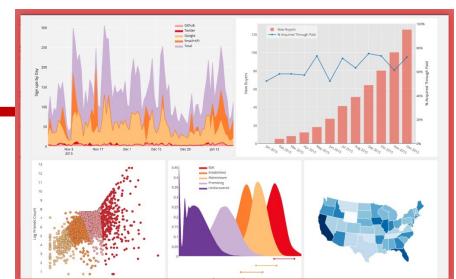
Reading and Processing Data



Data Structures and Algorithmic Thinking



Introduction to Object-Oriented Design



Creating Insightful Visualizations

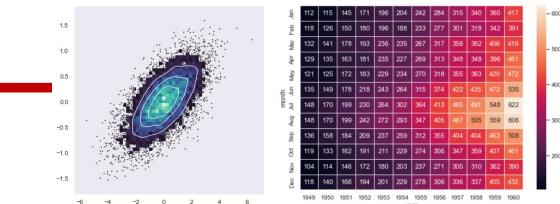
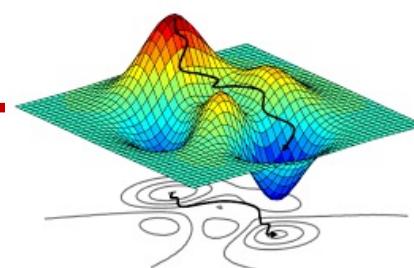
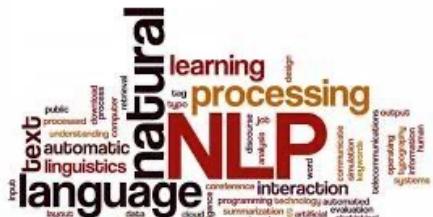


Northeastern University  
Khoury College of Computer Sciences

# DS 2500: Intermediate Programming with Data

- DS 2500 is the second course in Northeastern's *Programming with Data* sequence.
- DS 2500 is for DS majors, minors, and non-CS / non-DS majors who have taken DS 2000 or have equivalent programming experience.
- DS2500 is required for those wishing to eventually pursue more advanced data science courses such as DS3000 or DS3500.
- *DS 2500 is the course to take if you want to explore a broad range of data science topics using python.*

**Some of the topics usually explored in DS2500:**



Natural Language  
Processing (NLP)

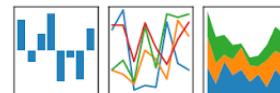
Graphs and Networks:  
Data structures and Analysis

Optimization and  
Machine Learning

Visualization Techniques



pandas  
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

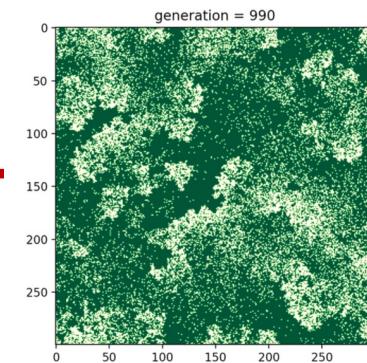
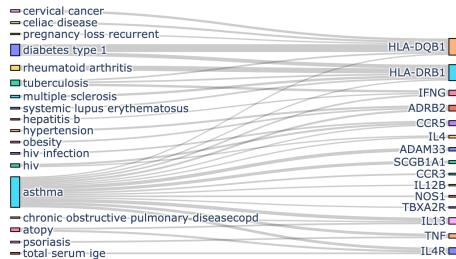


Northeastern University  
Khoury College of  
Computer Sciences

# DS 3500: Advanced Programming with Data

- DS 3500 is the third and final course in Northeastern's *Programming with Data* sequence.
- Requires DS2500: *Intermediate Programming with Data*
- DS 3500 is for DS majors, minors, and non-CS / non-DS majors who have taken DS 2500 or have equivalent programming experience.
- DS 3500 is the course to take if you want to become a professional software developer or data scientist.

Some of the topics we will explore this Fall 2022:



Building Interactive  
Visualizations & Dashboards

Working with relational  
and non-relational databases

Object-oriented and  
functional paradigms.

Animation, Simulation,  
and Modeling



Northeastern University  
Khoury College of  
Computer Sciences

# DS3500 Topics

---

## Tooling

- Managing environments
- Anaconda library management
- The *git* software control system
- The JetBrains PyCharm IDE

## Techniques

- Exception handling
- Object-oriented design patterns
- Test-Driven Development
- Functional programming
- Decorator patterns / Profiling
- Type hints
- Extensibility

## Modules

- Advanced visualization with `plot.ly`
- Modeling and simulation / Animation
- Risk analysis for business forecasting
- Building interactive dashboards
- Databases / Persistence
- Optimization and Intelligent Decision Support
- Evolutionary Computing
- Scientific computing / multi-core computing
- Working with streaming data
- Time-series analysis
- Data structures: Property graphs for Network Analysis
- Image processing
- Concurrency with `asyncio`
- Web-development / User-interfaces with `flask`
- Other?



DS3500 –Programming with Data

---

# Setting up your Python development environment



Northeastern University

# Three critical elements

---

**Anaconda:** A python distribution and environment manager (or “package manager”) that comes pre-installed with a *base* environment containing hundreds of packages. (Mine currently has 278!)

An **IDE** (Integrated Development Environment): For creating projects and writing, running, and testing code. We’ll be mostly using **Jetbrains PyCharm** in class but I might sometimes revert to spyder or Jupyter Notebooks. PyCharm readily integrates with Anaconda environments.

**GIT:** A software control system – for working together on shared development projects. PyCharm integrates with GIT.



# Python Distributions

---

A python *distribution* typically consists of:

- The python programming language
- Development tools
- Pre-installed python packages (libraries) that make python more powerful
- A package manager (allows you to customize your distribution by adding more libraries or updating existing libraries)





» PythonDistributions

» PythonDistributions

## Python Distributions

Aside from the official CPython distribution available from python.org, other distributions based on CPython include the following:

- »  [ActivePython from ActiveState](#)
- »  [Anaconda from Continuum Analytics](#)
- »  [ChinesePython Project](#): Translation of Python's keywords, internal types and classes into Chinese. Eventually allows a program to run on both English and Chinese environments.
- »  [Enthought's Canopy](#)
- »  [Win9xPython](#): Backport of mainline CPython 2.6/2.7 to old versions of Windows 9x/NT.
- »  [IPython](#) and its  [IPyKit](#) variant
- »  [PocketPython](#)
- »  [Portable Python](#): Run Python from USB device - no installation needed.
- »  [PyIMSL Studio](#)
- »  [PyPy](#): a Python implementation in Python.
- »  [Python\(x,y\)](#): Python(x,y) is a scientific-oriented Python Distribution based on Qt, Eclipse and  [Spyder](#)
- »  [PythonForArmLinux](#)
- »  [PythonLabsPython](#): an old name for the python.org distribution
- »  [PythonwarePython](#)
- »  [StacklessPython](#)
- »  [Tiny Python](#) (archived link) - not to be confused with  [tinypy](#)
- »  [WinPython](#): Another scientific-focused Python distribution, based around  [Spyder](#)

There are other python distributions out there, but we'll use Anaconda.



Northeastern University

[www.anaconda.com/download](http://www.anaconda.com/download)

---

**Anaconda Distribution**

# Free Download

Everything you need to get started in data science on your workstation.

- ✓ Free distribution install
- ✓ Thousands of the most fundamental DS, AI, and ML packages
- ✓ Manage packages and environments from desktop application
- ✓ Deploy across hardware and software platforms

 [Code in the Cloud](#)

 [Download](#) ▾

[Get Additional Installers](#)

 |  | 



Northeastern University

# Anaconda installers

<https://www.anaconda.com/download#downloads>

(Skip Registration if you want.)

## Anaconda Installers



### Windows

#### Python 3.12

⤵ 64-Bit Graphical Installer (912.3M)



### Mac

#### Python 3.12

⤵ 64-Bit (Apple silicon) Graphical  
Installer (704.7M)

⤵ 64-Bit (Apple silicon) Command  
Line Installer (707.3M)

⤵ 64-Bit (Intel chip) Graphical  
Installer (734.7M)

⤵ 64-Bit (Intel chip) Command Line  
Installer (731.2M)



### Linux

#### Python 3.12

⤵ 64-Bit (x86) Installer (1007.9M)

⤵ 64-Bit (AWS Graviton2 / ARM64)  
Installer (800.6M)

⤵ 64-bit (Linux on IBM Z & LinuxONE)  
Installer (425.8M)



Northeastern University

# Installing Anaconda on a Mac



Anaconda3-2023.07-2-MacOSX-x86\_64.pkg

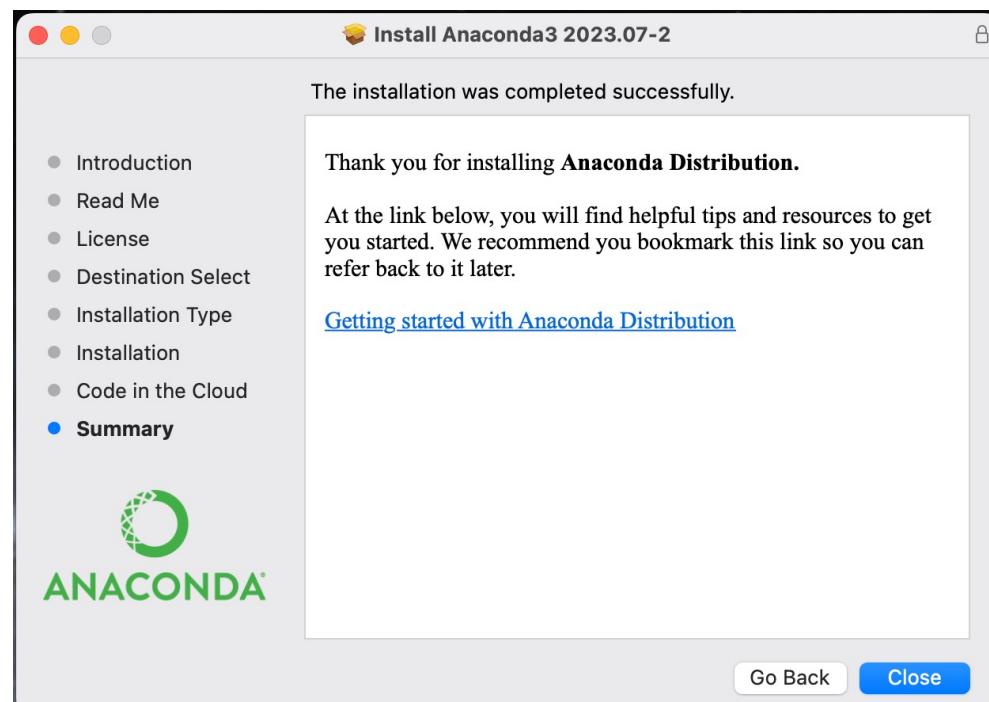
Completed — 611 MB

I chose /Users/rachlin/opt as my installation folder. The home directory for anaconda is then:

**/Users/rachlin/opt/anaconda3**

It installs Anaconda Navigator and tries to get you to set up a cloud account. I don't use it.

The "base" environment uses python 3.11.4



Northeastern University

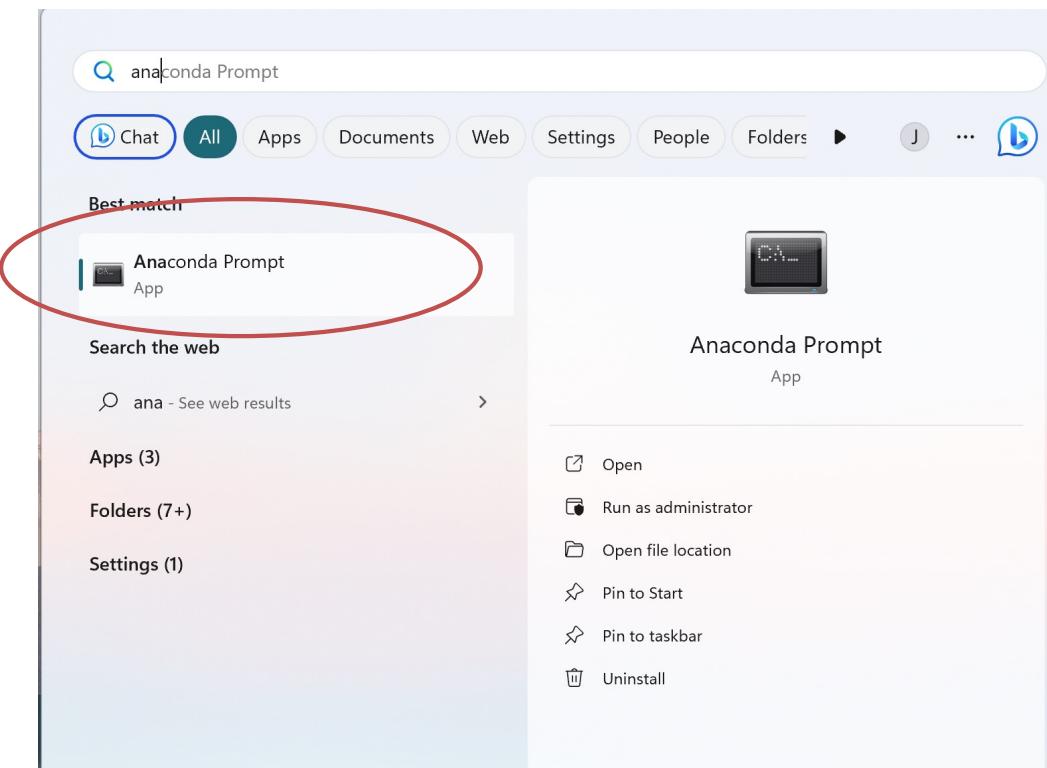
# Anaconda on Windows

I just accepted defaults or any recommended buttons when installing on windows.



Northeastern University

# Running conda commands on Windows



To run “conda” commands on windows, use the Anaconda Prompt. (You might want to pin this application to your Task Bar.)

It basically opens a command terminal with the base anaconda environment activated.



# conda info

Note that the base environment lives in the anaconda home directory. Other environments live in an **envs** subdirectory.

```
(base) [501 uranus:~] conda info

active environment : base
active env location : /Users/rachlin/opt/anaconda3
shell level : 1
user config file : /Users/rachlin/.condarc
populated config files : /Users/rachlin/.condarc
conda version : 23.7.2
conda-build version : 3.26.0
python version : 3.11.4.final.0
virtual packages : __archspec=1=x86_64
__osx=10.16=0
__unix=0=0
base environment : /Users/rachlin/opt/anaconda3 (writable)
conda av data dir : /Users/rachlin/opt/anaconda3/etc/conda
conda av metadata url : None
channel URLs : https://repo.anaconda.com/pkgs/main/osx-64
https://repo.anaconda.com/pkgs/main/noarch
https://repo.anaconda.com/pkgs/r/osx-64
https://repo.anaconda.com/pkgs/r/noarch
package cache : /Users/rachlin/opt/anaconda3/pkgs
/Users/rachlin/.conda/pkgs
envs directories : /Users/rachlin/opt/anaconda3/envs
/Users/rachlin/.conda/envs
platform : osx-64
user-agent : conda/23.7.2 requests/2.31.0 CPython/3.11.4 Darwin/22.6.0 OSX/10.16
UID:GID : 501:20
netrc file : None
offline mode : False
```



# IPython: An enhanced REPL for Python

```
● ● ● rachlin — IPython: Users/rachlin — ipython — 80x24
(base) [512 uranus:~] ipython
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

[In 1]: print('hi from IPython')
hi from IPython

[In 2]: ?print
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type: builtin_function_or_method

[In 3]:
```

Syntax highlighting  
Code completion  
Documentation

You'll want to install IPython in your **ds** environment. It's useful.



# Create a “ds” environment

---

The base environment comes with almost 500 libraries, most of which you don’t need. Run the following commands to set up a **ds** environment with a few starting python libraries.

```
conda create --name ds python=3.11
conda activate ds
conda install matplotlib
conda install pandas
conda install ipython
```

**Checking that I have two environments, and where they are located**

```
conda env list
```

```
# conda environments:
#
base            /Users/rachlin/opt/anaconda3
* ds            /Users/rachlin/opt/anaconda3/envs/ds
```



# Environment locations on Windows

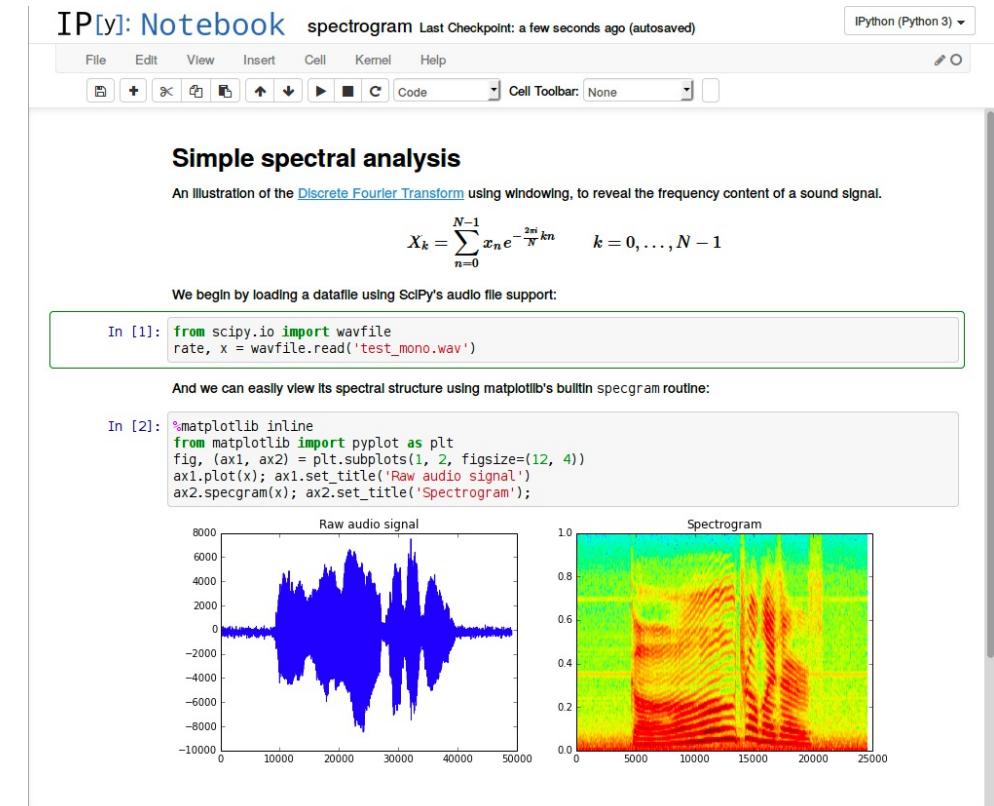
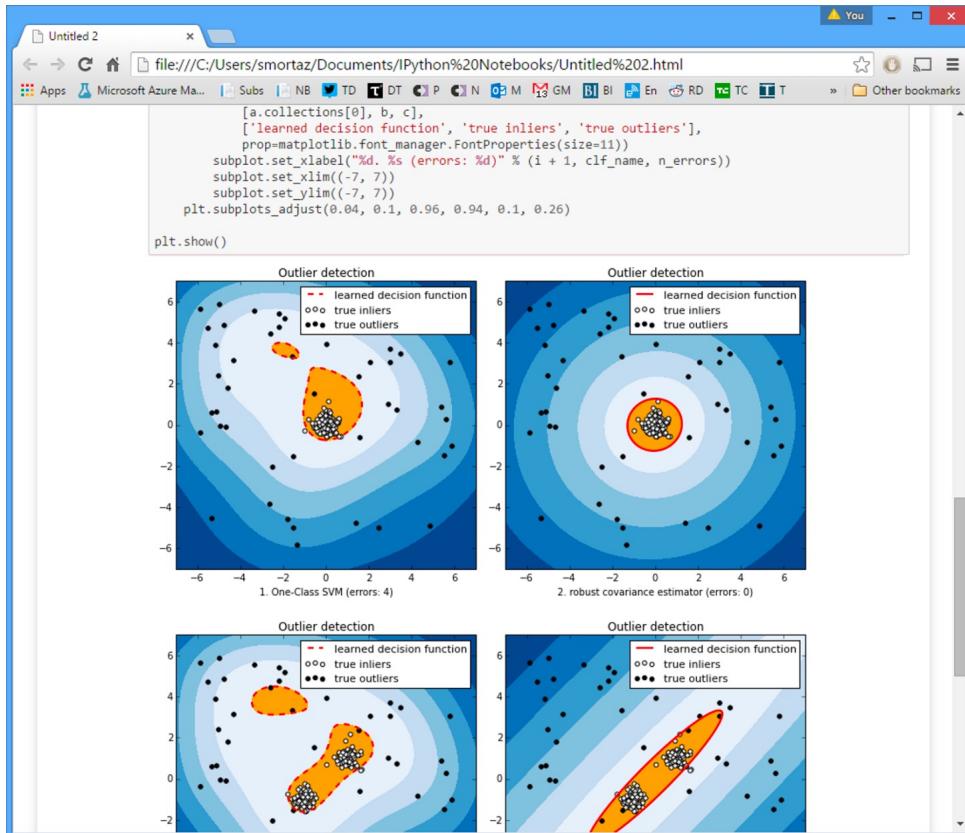
---

On windows, it is the same (but I have a different username).

```
(ds) C:\Users\johnr>conda env list
# conda environments:
#
base          C:\Users\johnr\anaconda3
*  ds           C:\Users\johnr\anaconda3\envs\ds
```



# Jupyter Notebooks: Interactive Python for Data Science

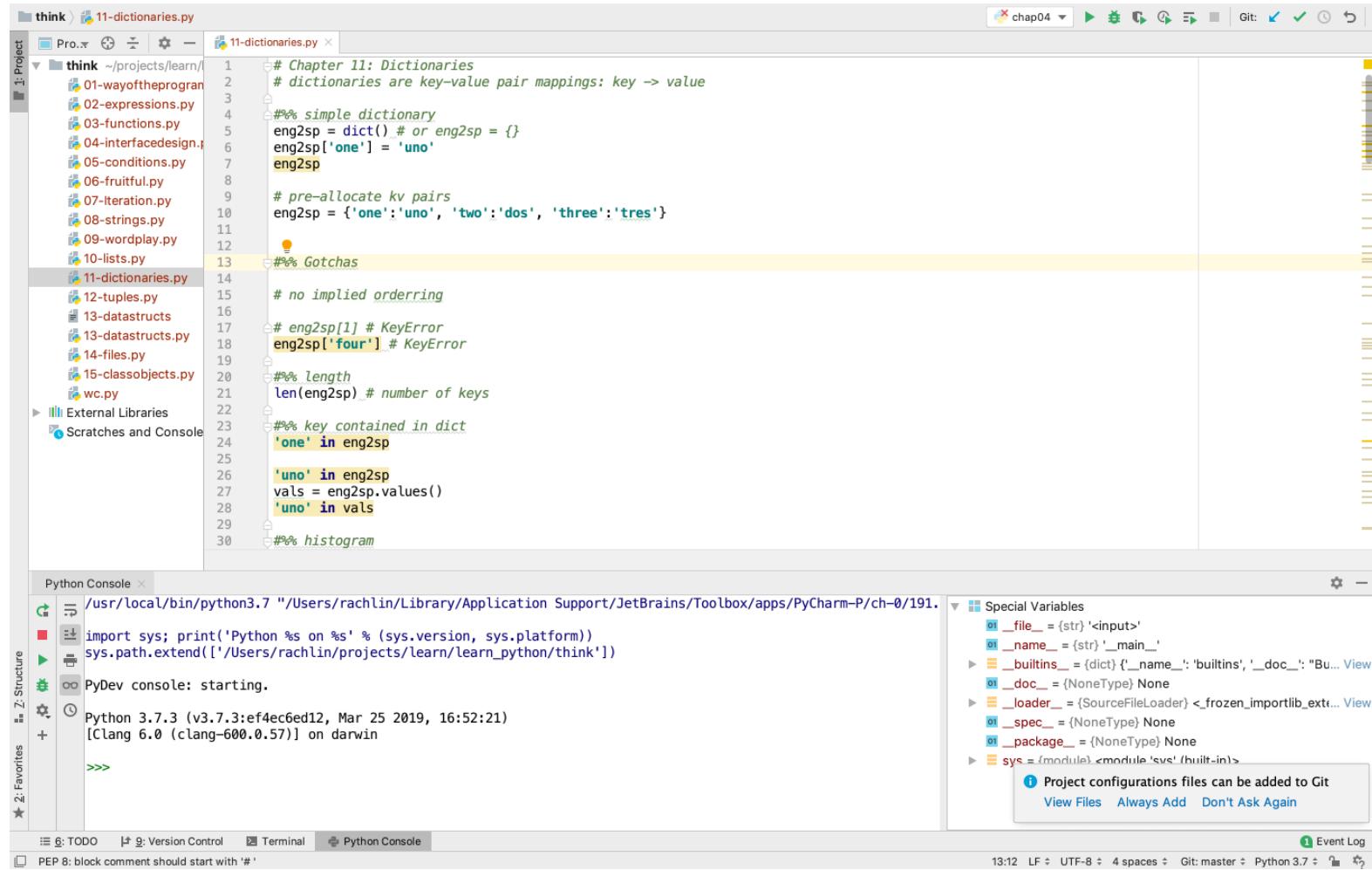


Jupyter is useful for data science but not for the types of programs we'll be writing in this class where we focus on code reuse by building *libraries*.



Northeastern University

# PyCharm (JetBrains): Recommended Python IDE



# PyCharm: Community or Professional?

---

<https://www.jetbrains.com/shop/eform/students>

## JetBrains Products for Learning

---

Before you apply, please read the [Educational Subscription Terms and FAQ](#).

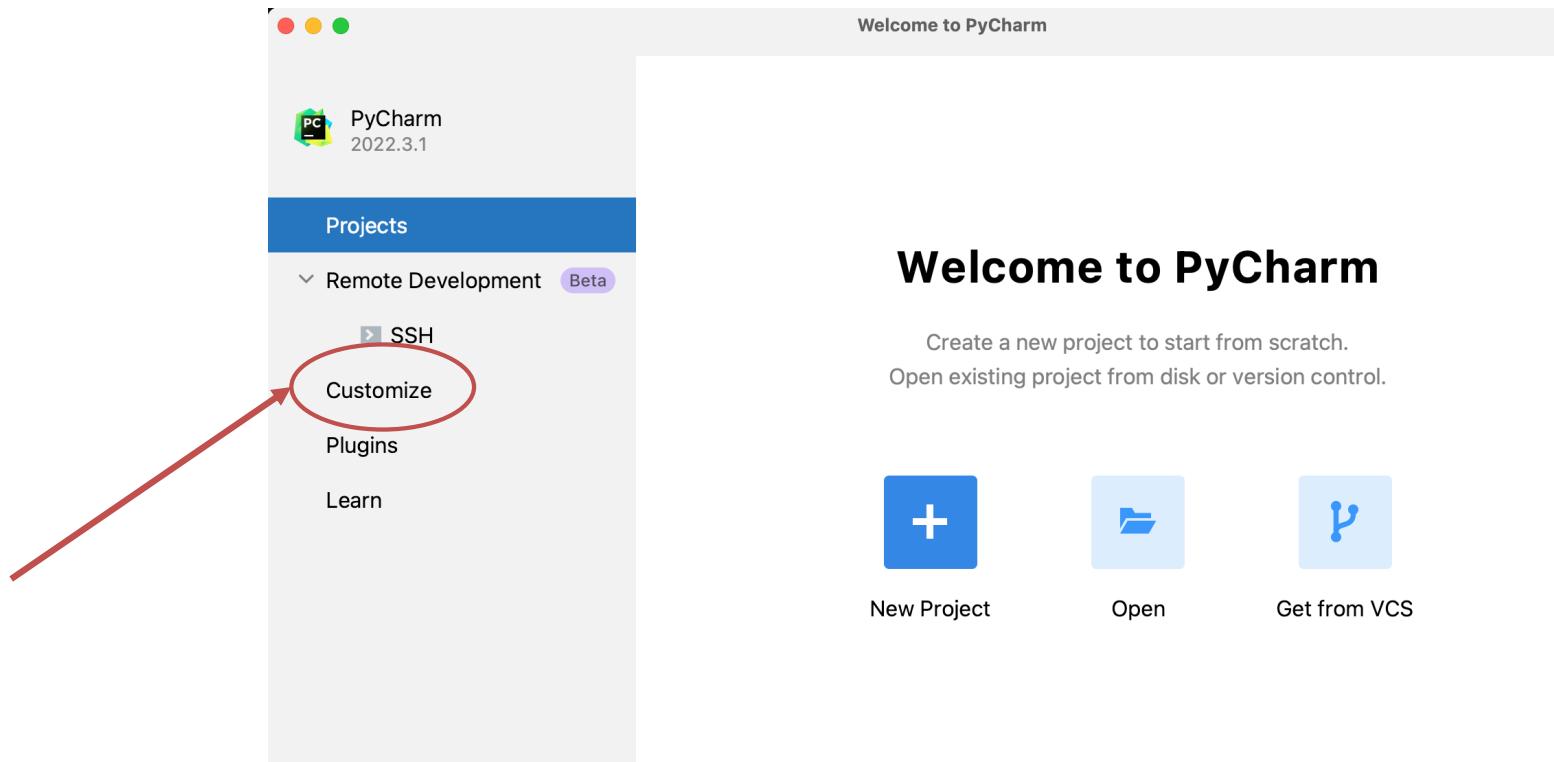
The screenshot shows a web form for applying for a JetBrains educational license. At the top, there are four options for 'Apply with': 'University email address' (which is selected and underlined in blue), 'ISIC/ITIC membership', 'Official document', and 'GitHub'. Below this, the 'Status:' section has two radio buttons: 'I'm a student' (selected) and 'I'm a teacher'. The 'Level of study' section contains a dropdown menu set to 'Undergraduate'. At the bottom, a question asks 'Is Computer Science or Engineering your major field of study?' with two radio buttons: 'Yes' (selected) and 'No'.

The community edition should be fine for this class, but you can get access to the professional edition and other JetBrains products for free by signing up for an Educational License (Free).

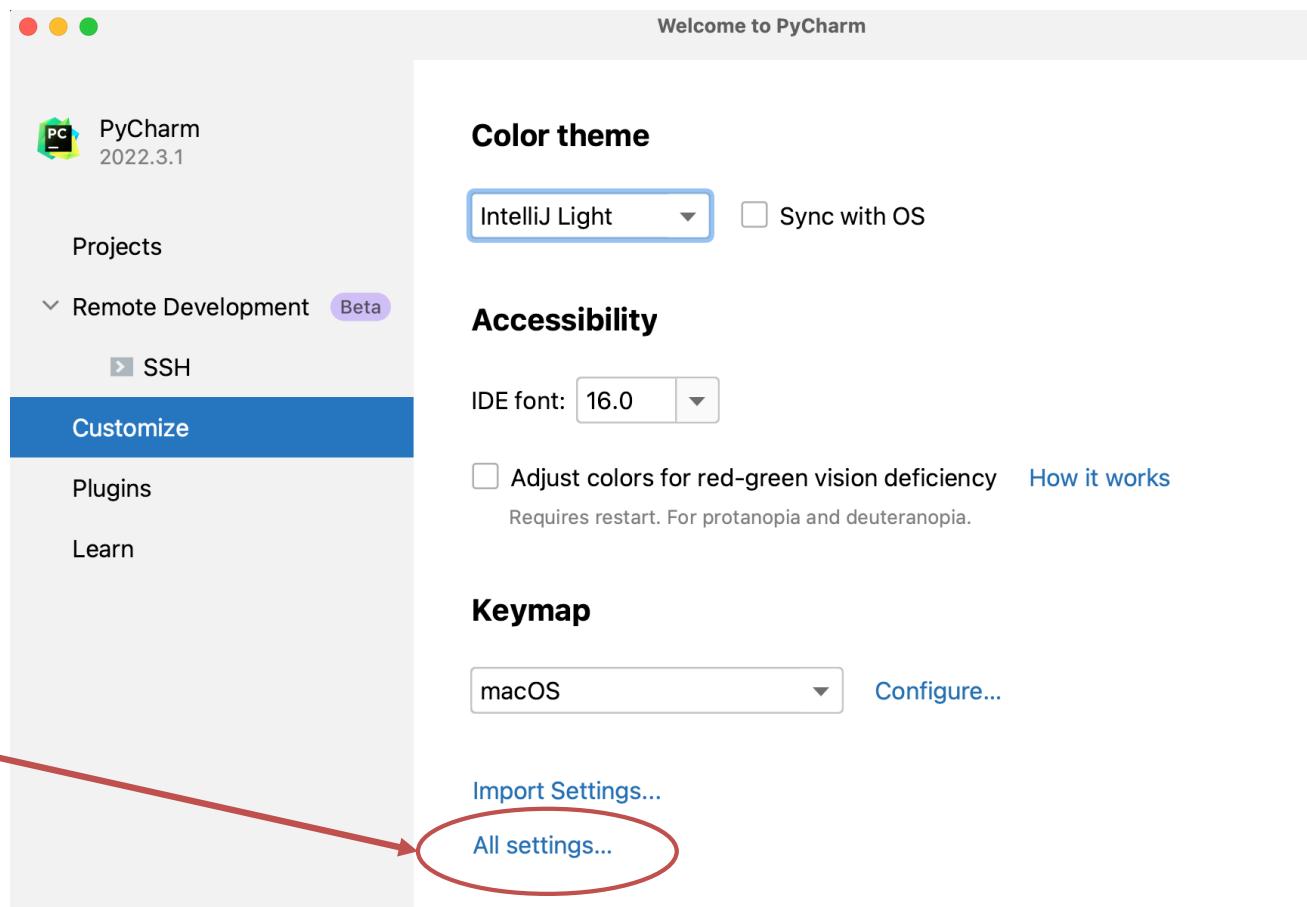


Northeastern University

# Connecting PyCharm project to an environment

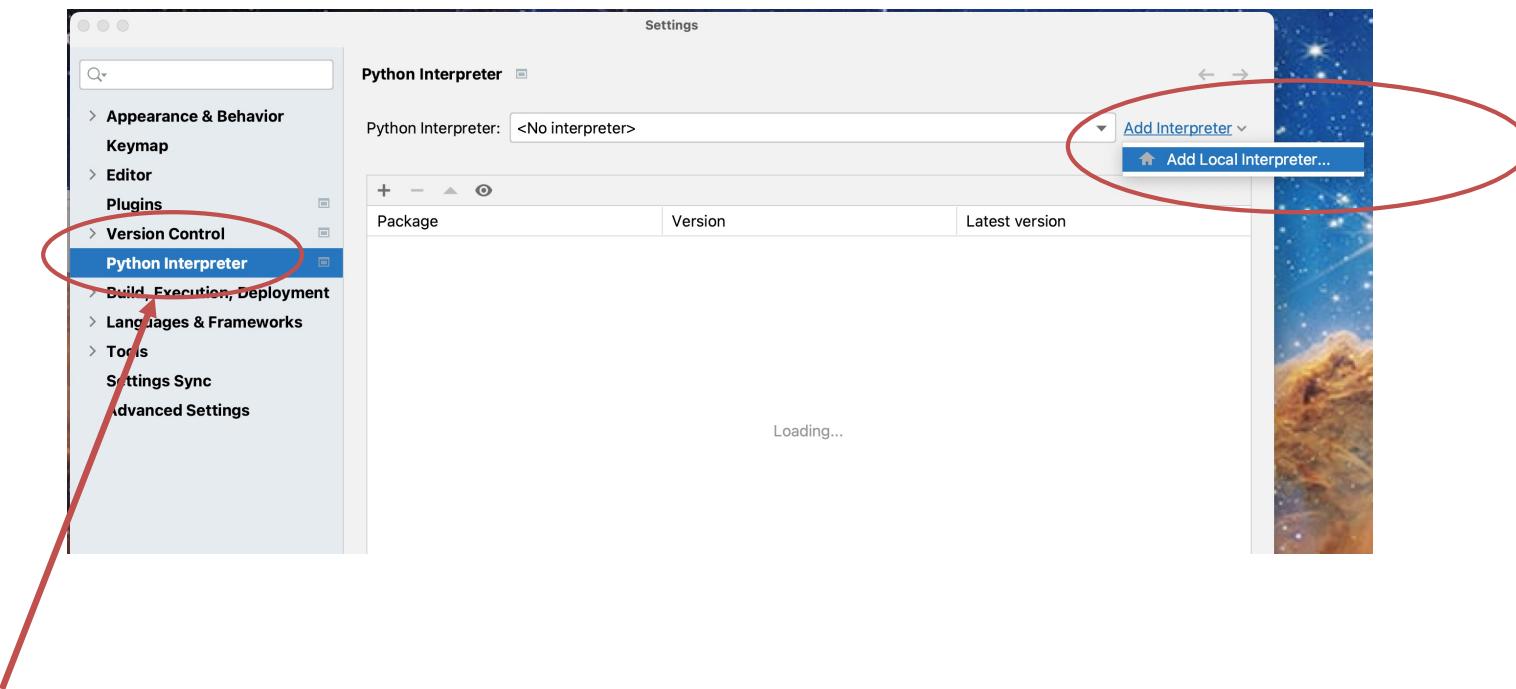


# Connecting PyCharm project to an environment

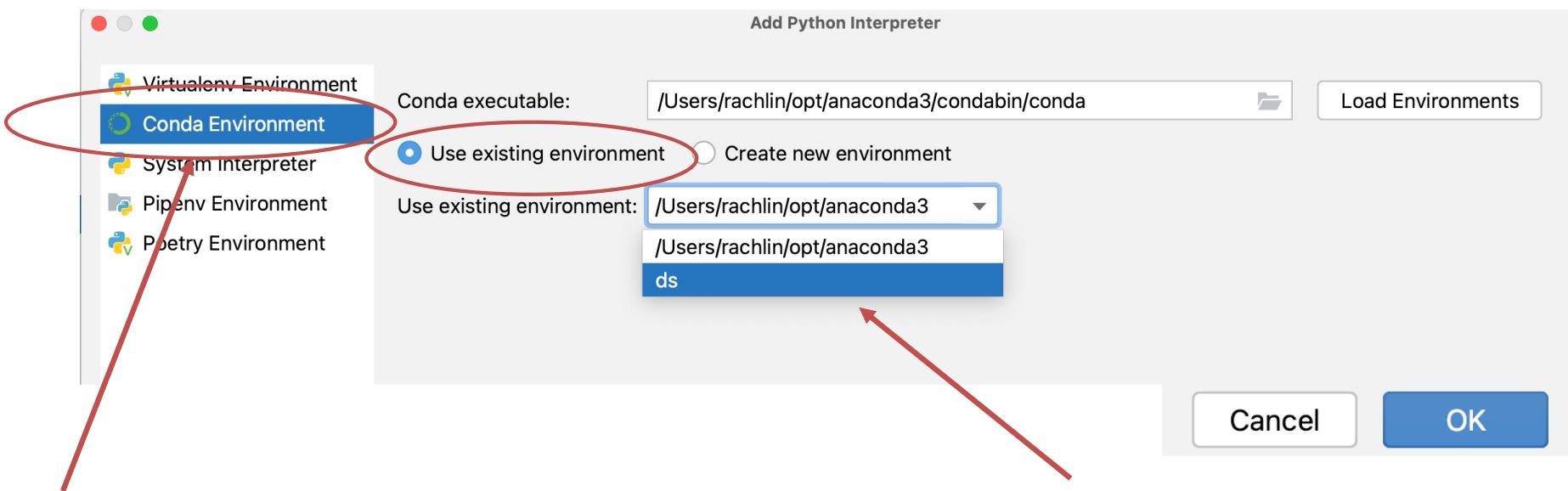


Northeastern University

# Connecting PyCharm project to an environment



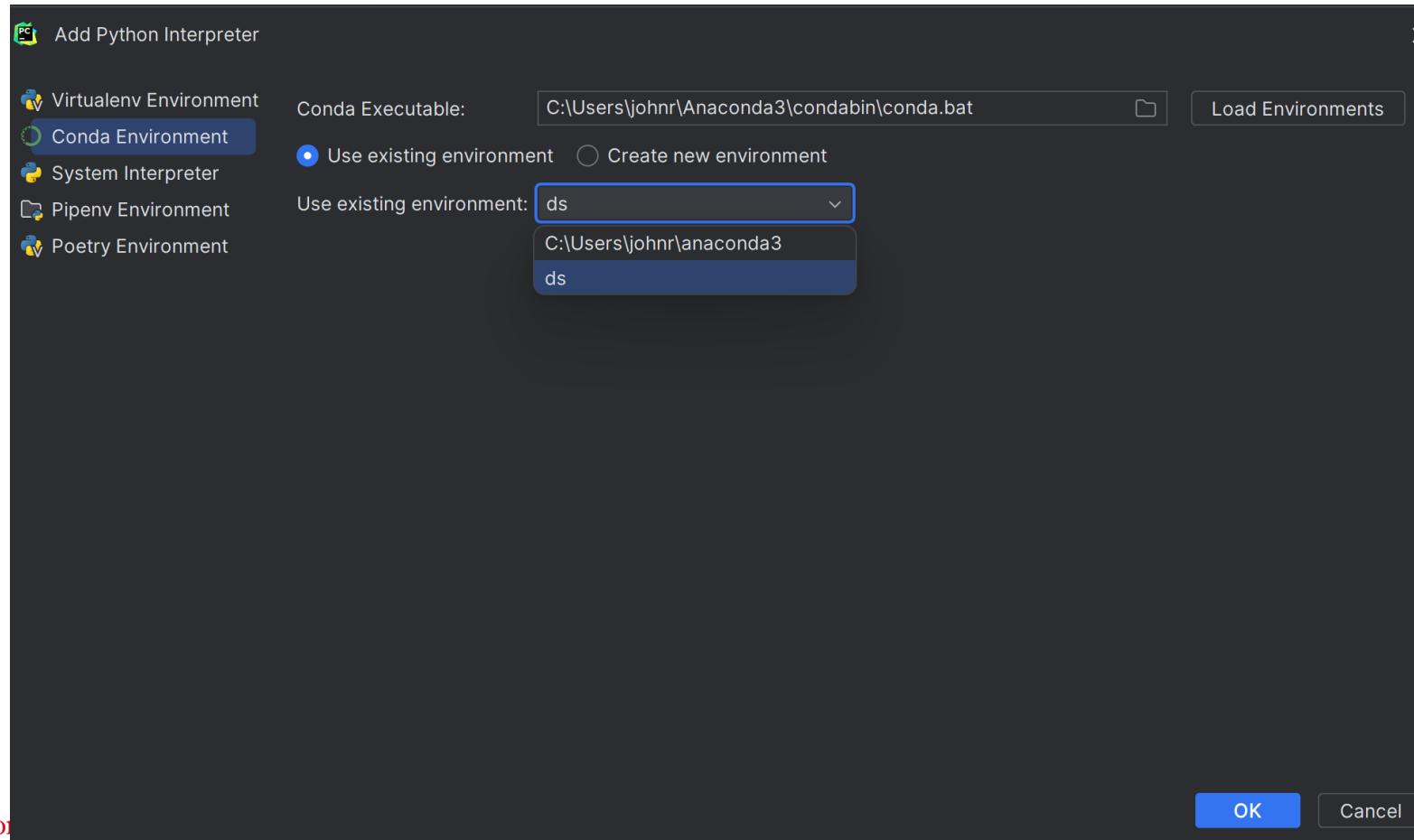
# Connecting PyCharm project to an environment



Select the “ds” environment you created and then press OK



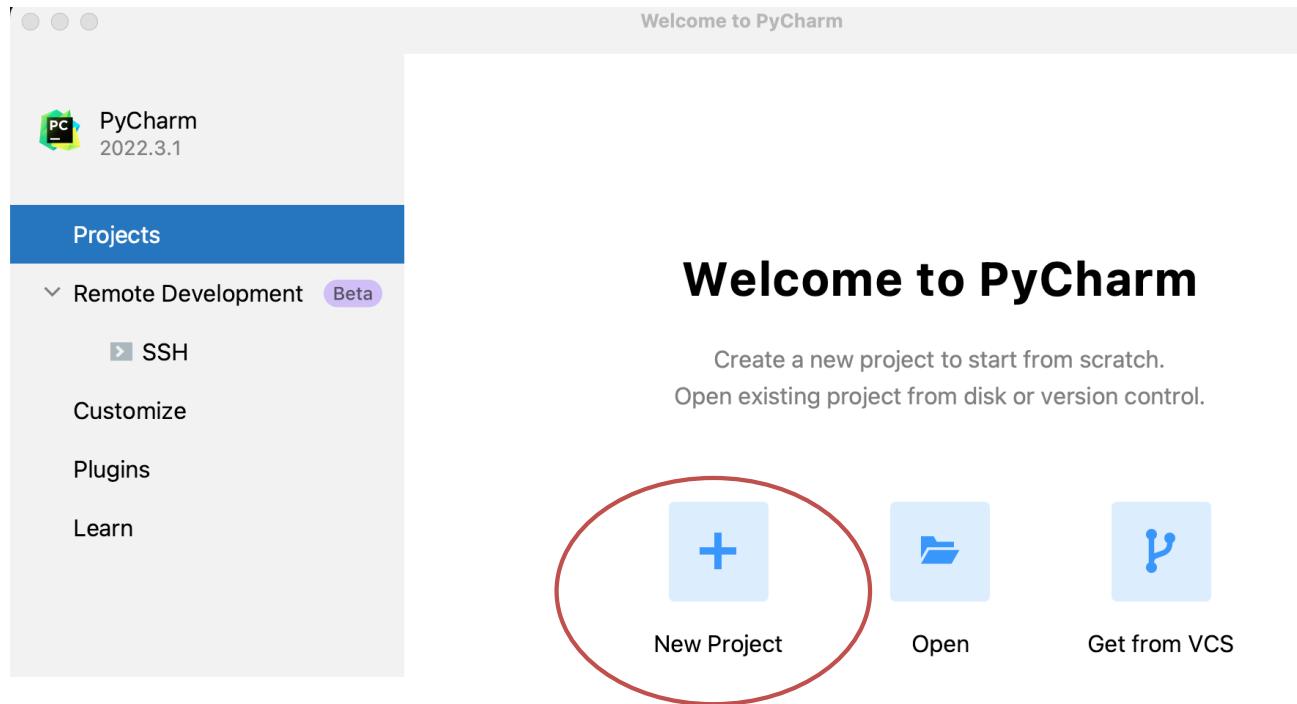
# Very similar in Windows



No

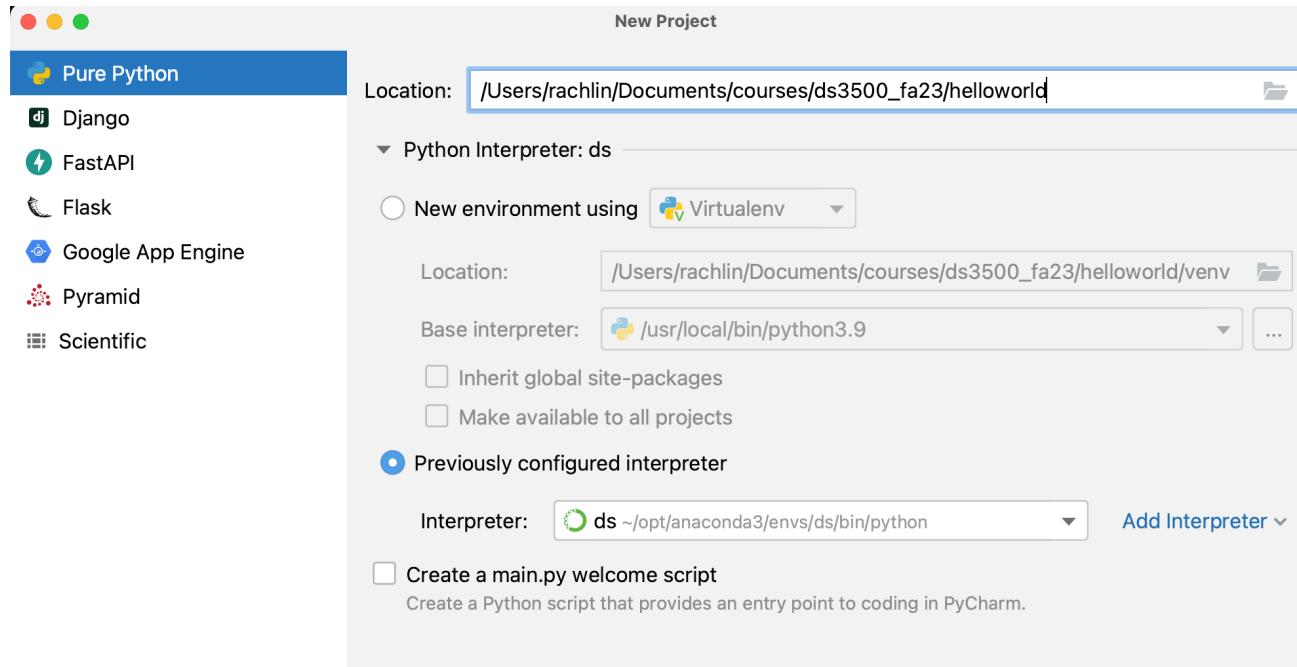
# Create your first project

---



Northeastern University

# Create your first project

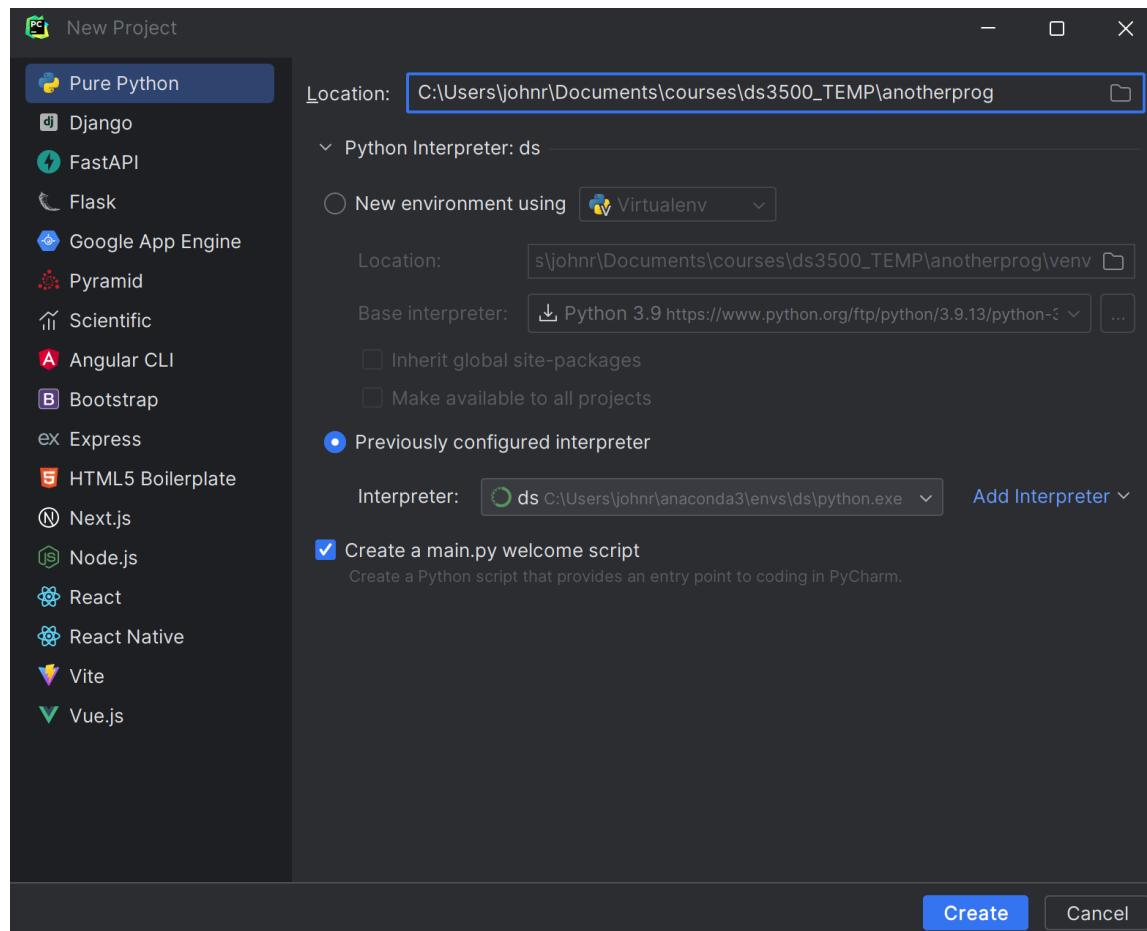


Here I'm creating a project folder in my local copy of the class repo (ds3500\_fa23).  
You would create YOUR projects and code outside the repo folders – perhaps in your OWN repo!



Northeastern University

# Windows: Creating a New Project (same)

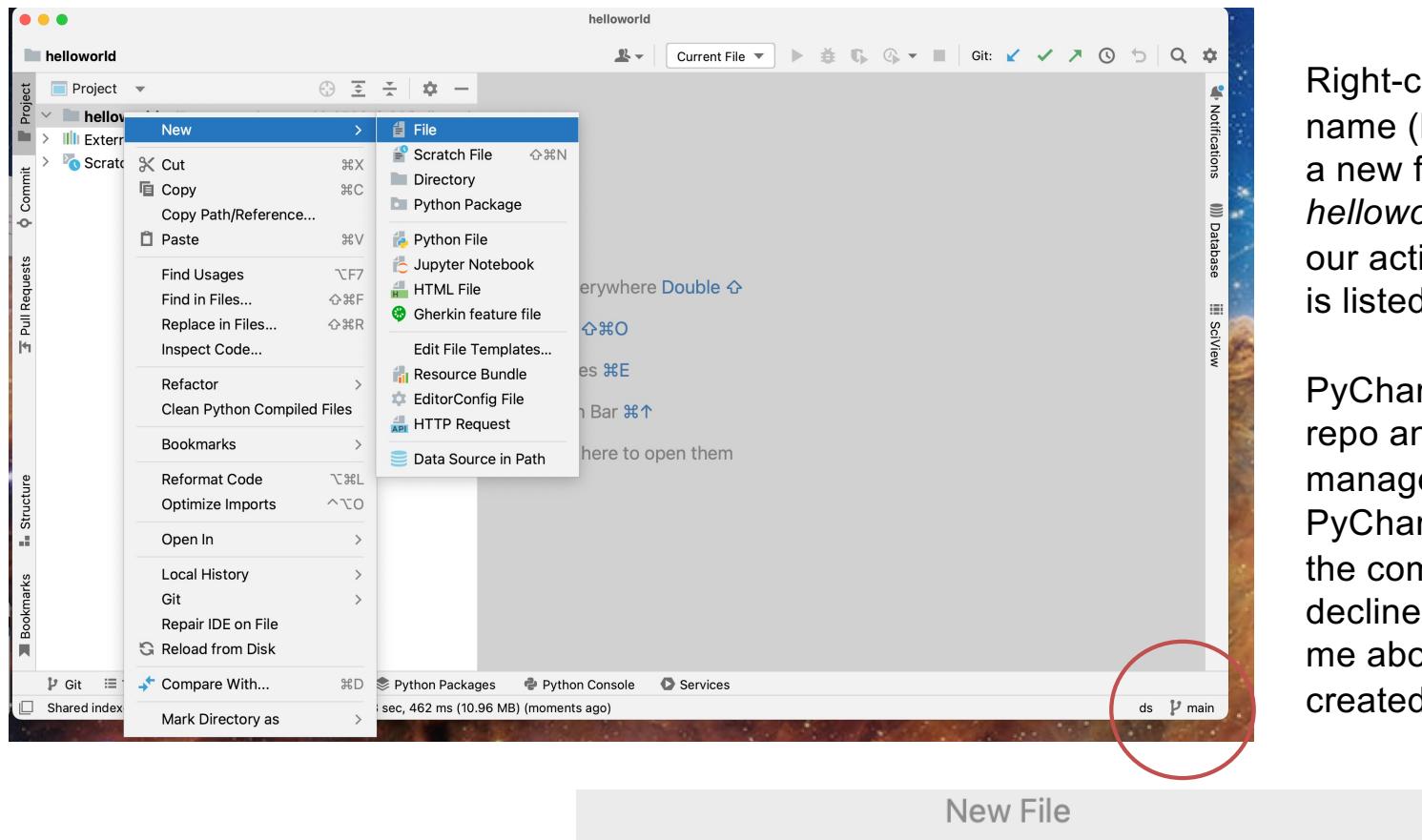


I created this project  
in a different repo / folder:  
ds3500\_TEMP



Northeastern University

# Creating our first python program

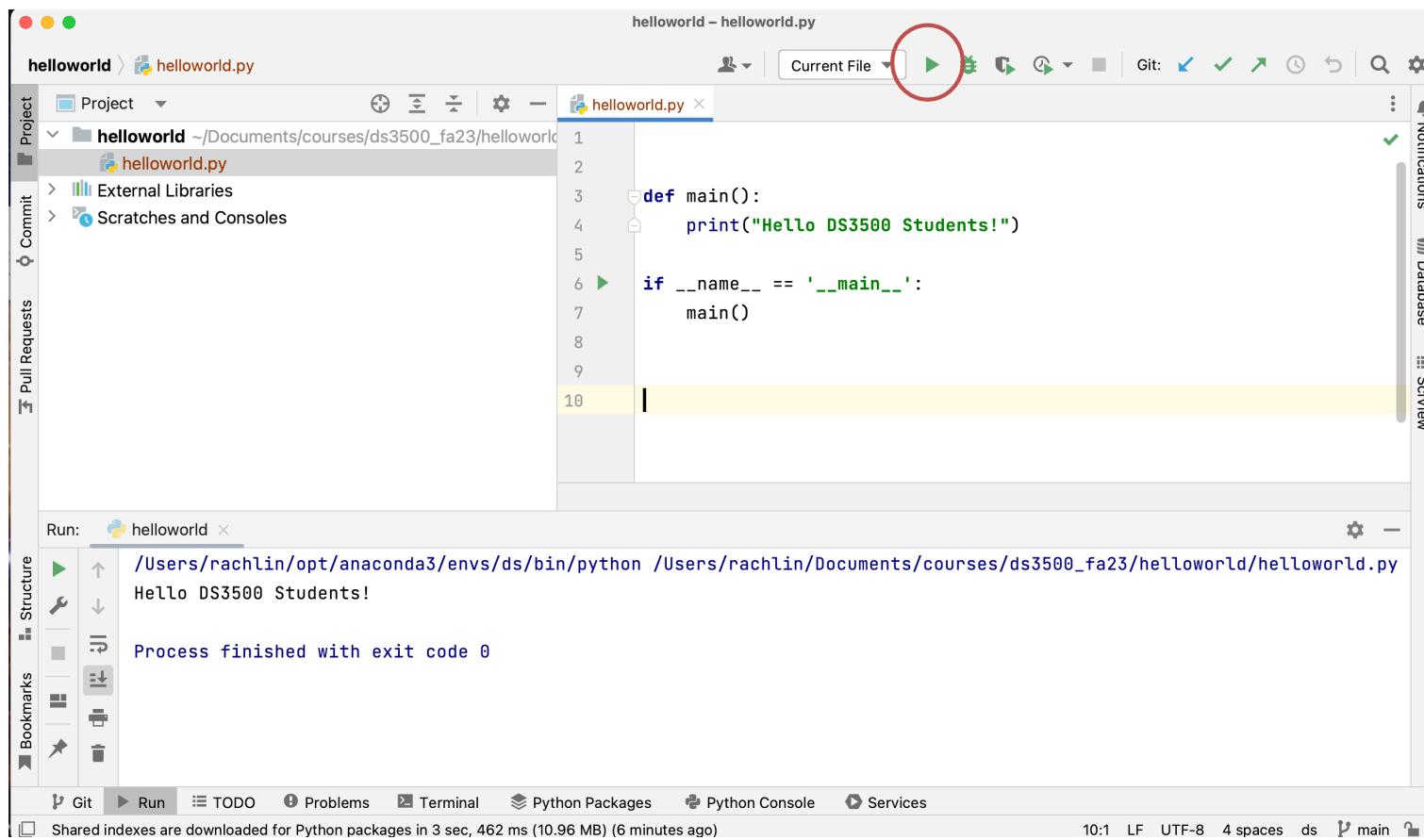


Right-clicking on the project name (**helloworld**) and adding a new file which will be called *helloworld.py*. Notice that “ds”, our active environment is listed in the status bar.

PyCharm knows this is a git repo and it is possible to manage your repo through PyCharm. I prefer to use the command line, so I will decline when PyCharm asks me about tracking newly created files.



# Running our program



The screenshot shows a Python development environment with the following details:

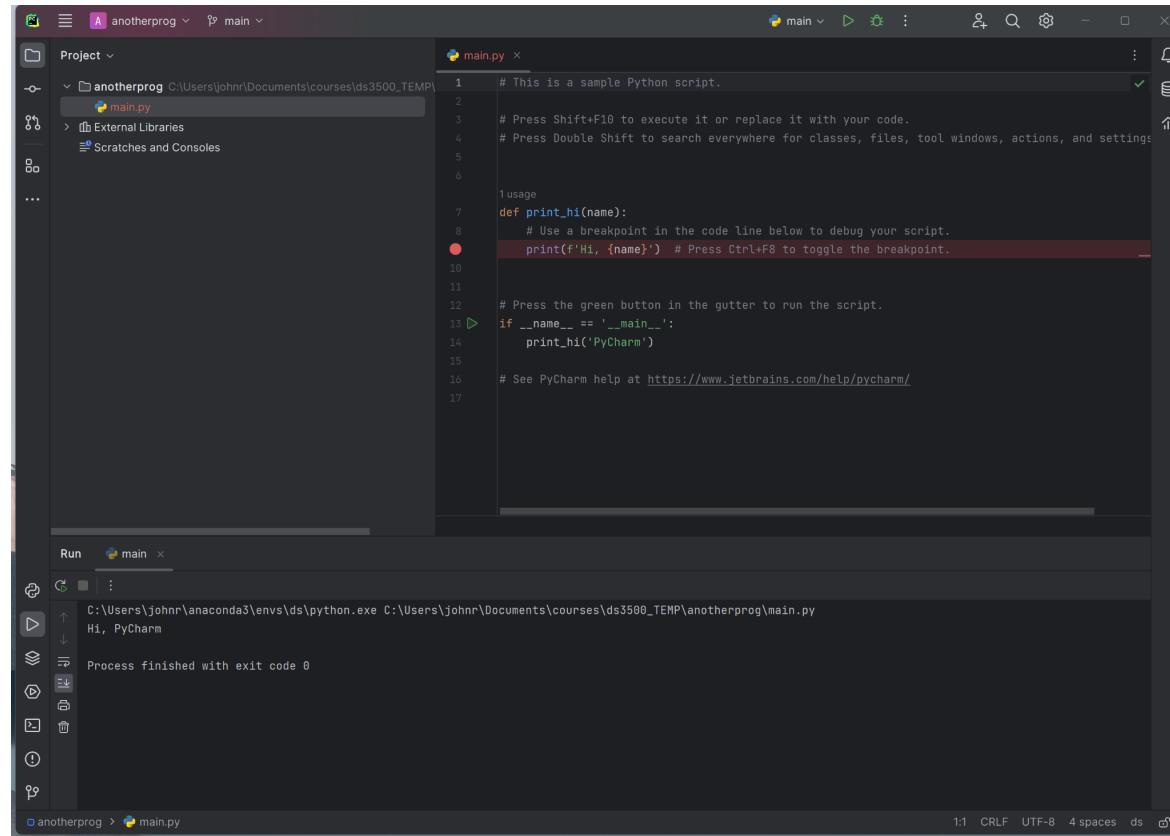
- Project:** helloworld (~/Documents/courses/ds3500\_fa23/helloworld)
- File:** helloworld.py
- Code:**

```
1
2
3 def main():
4     print("Hello DS3500 Students!")
5
6 if __name__ == '__main__':
7     main()
8
9
10
```
- Toolbar:** Includes a red circle highlighting the green run button.
- Run Tab:** Shows the command run: /Users/rachlin/opt/anaconda3/envs/ds/bin/python /Users/rachlin/Documents/courses/ds3500\_fa23/helloworld/helloworld.py and the output:

```
Hello DS3500 Students!
Process finished with exit code 0
```
- Bottom Status Bar:** Shared indexes are downloaded for Python packages in 3 sec, 462 ms (10.96 MB) (6 minutes ago), 10:1 LF UTF-8 4 spaces ds main



# Windows Interface (dark theme)



You can choose different themes and syntax highlighting schemes.



# Python Development Tools - Summary

Tool / Platform	Strengths and Weaknesses
IPython (RECOMMENDED)	Great for testing small snippets of Python and for learning Python syntax. Not intended for application development. IPython is an enhanced REPL for Python that comes packaged with the Anaconda distribution.
IDLE (NOT FOR ADVANCED)	Start coding today! Comes with Python distribution. Fine for small, single-file, stand-alone scripts or applications. Not recommended for more complex projects.
Atom text editor (OBSOLETE)	An extendible text-editor with many plugins for Python programming. See Prof. Rachlin's handout on recommended plugins.
Spyder (LIMITED)	A full-fledged IDE installed with the Anaconda distributions. Manage multiple files. Monitor variables. Run isolated blocks of code.
PyCharm (JetBrains) (RECOMMENDED)	Another popular IDE for python. Has more functionality than what you will need for this class. I won't be making use of PyCharm in this class, but you are welcome to explore its capabilities and use it for your homework.
Jupyter notebooks (PROTOTYPING ONLY)	A <i>notebook-style</i> development environment popular with data scientists. Integrate code, text, tables, visualizations in one file. Not suited for software <i>applications</i> requiring multiple python files.
Anaconda (REQUIRED)	A Python distribution that comes with Python, popular pre-installed libraries, and a variety of development tools including Spyder, Ipython, and Jupyter notebooks. The Anaconda Navigator is a jumping-off point for applications, learning resources, and community forums. <b>Recommended.</b>

# Installing Git on a Mac

The Git Pro books has instructions: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

There are several ways to install Git on macOS. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run `git` from the Terminal the very first time.

```
$ git --version
```

If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is maintained and available for download at the Git website, at <https://git-scm.com/download/mac>.



Northeastern University

# Installing Git on windows.

The Git Pro books has instructions: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

The screenshot shows the official Git website ([git-scm.com](https://git-scm.com/)) with a focus on the Windows download page. The header features the Git logo (a red diamond with a white 'g') and the tagline "git --local-branching-on-the-cheap". A search bar is located in the top right corner. On the left side, there's a sidebar with links for "About", "Documentation", "Downloads" (which is highlighted in red), and "Community". Below the sidebar, a box contains text about the "Pro Git book" by Scott Chacon and Ben Straub being available online for free on Amazon.com. The main content area is titled "Download for Windows" and provides a link to download the latest 64-bit version (2.42.0). It also lists other download options: "Standalone Installer", "32-bit Git for Windows Setup.", "64-bit Git for Windows Setup.", "Portable ("thumbdrive edition")", "32-bit Git for Windows Portable.", and "64-bit Git for Windows Portable.".

**git** --local-branching-on-the-cheap

Search entire site...

**About**

**Documentation**

**Downloads**

GUI Clients

Logos

**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on [Amazon.com](#).

## Download for Windows

[Click here to download](#) the latest (2.42.0) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **15 days ago**, on 2023-08-30.

**Other Git for Windows downloads**

[Standalone Installer](#)

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

[Portable \("thumbdrive edition"\)](#)

[32-bit Git for Windows Portable.](#)

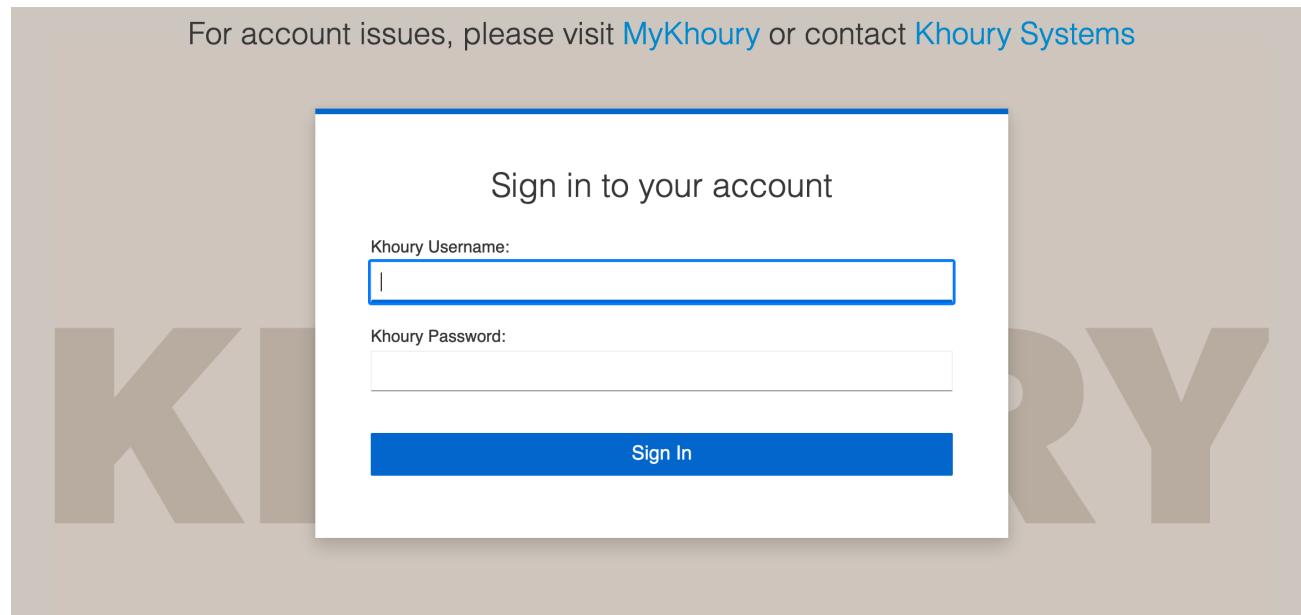
[64-bit Git for Windows Portable.](#)



# The Khoury Enterprise GIT Server

---

<https://github.khoury.northeastern.edu/>



Northeastern University

# Post-git Installation Commands

---

## Your Identity

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com
```

Again, you need to do this only once if you pass the `--global` option, because then Git will always use that information for anything you do on that system. If you want to override this with a different name or email address for specific projects, you can run the command without the `--global` option when you're in that project.

Many of the GUI tools will help you do this when you first run them.



# Post-git Installation Commands

---

## Your Editor

Now that your identity is set up, you can configure the default text editor that will be used when Git needs you to type in a message. If not configured, Git uses your system's default editor.

If you want to use a different text editor, such as Emacs, you can do the following:

```
$ git config --global core.editor emacs
```

On a Windows system, if you want to use a different text editor, you must specify the full path to its executable file. This can be different depending on how your editor is packaged.

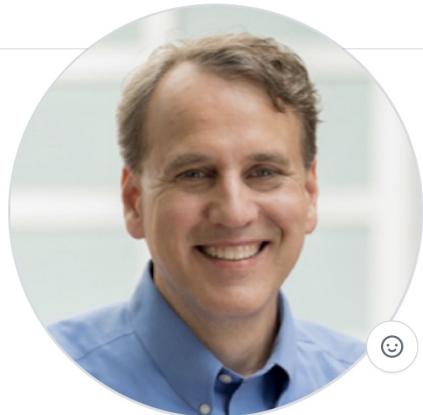
In the case of Notepad++, a popular programming editor, you are likely to want to use the 32-bit version, since at the time of writing the 64-bit version doesn't support all plug-ins. If you are on a 32-bit Windows system, or you have a 64-bit editor on a 64-bit system, you'll type something like this:

```
$ git config --global core.editor "C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
```



# GIT Repository List

Why is my  
face so big?



**rachlin**  
rachlin

Associate Teaching Professor Khouri  
College of Computer Sciences  
Northeastern University

[Edit profile](#)



Northeastern University

Overview    Repositories    Projects    Stars

Find a repository...    Type ▾    Language ▾    Sort ▾    New

**ds3500\_fa23** Public  
Official Class Repo for DS3500 (Fall 2023)  
13 Updated 6 hours ago

**ds3500\_sp23** Private  
Course Repository for DS3500 (Sp23)  
Jupyter Notebook Updated on Apr 7

**web** Private  
Khoury Website  
HTML Updated on Feb 7

A screenshot of a GitHub repository list page. The top navigation bar includes 'Overview', 'Repositories' (which is highlighted with a red underline), 'Projects', and 'Stars'. Below the navigation are search and filter fields ('Find a repository...', 'Type ▾', 'Language ▾', 'Sort ▾', 'New'). The first repository listed is 'ds3500\_fa23' (Public), described as the 'Official Class Repo for DS3500 (Fall 2023)'. It has 13 stars and was updated 6 hours ago. The second repository is 'ds3500\_sp23' (Private), described as the 'Course Repository for DS3500 (Sp23)', containing a Jupyter Notebook and updated on April 7. The third repository is 'web' (Private), described as the 'Khoury Website', containing HTML and updated on February 7. A red circle highlights the 'New' button in the top right. A red line also highlights the 'Sort ▾' button.

# Creating a new repository

Here I'm creating a PRIVATE repo called ds3500\_TEMP.

I'll have a README file.

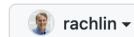
Setting the .gitignore template to Python means that when we check for new files to add to the repo, git will ignore certain files and folders related to working with python projects or with particular IDEs but that are not part of the code base.

## Create a new repository

A repository contains all project files, including the revision history.

Owner \*

Repository name \*



rachlin

/ ds3500\_TEMP



Great repository names are short and memorable. Need inspiration? How about [didactic-waffle](#)?

Description (optional)

This is a temporary repo, just for demonstration purposes

Public

Any logged in user can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).



This will set `main` as the default branch. Change the default name in your [settings](#).

You are creating a private repository in your personal account.

**Create repository**



Northeastern University

# SSH Keys

Before I can clone my repo or run other git commands, I have to register my laptop with the Khoury Enterprise Git Server. **Go to Settings.... SSH and GPG Keys.**

The screenshot shows the GitHub 'SSH keys' settings page for the user 'rachlin'. On the left is a sidebar with navigation links like 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Emails', 'Password and authentication', 'Sessions', 'SSH and GPG keys' (which is highlighted), and 'Organizations'. The main area is titled 'SSH keys' and contains a list of three registered keys:

- Uranus Laptop**  
SHA256: s3XY2Z36p0P7lZW4hr0qyj0pAuP/gIRa0JsFtyrpSjk  
Added on Sep 10, 2022  
Last used within the last week — Read/write
- Mercury Linux**  
SHA256: hr9UVyGgKSrK9GJqsJ+XpnNp197p4cmHyi3rb02vz1I  
Added on Jan 21, 2023  
Last used within the last 2 weeks — Read/write
- Mercury Windows**  
SHA256: T8hlysgtZ/U1XUAF/0WHLhE3GB+Q1zWwpYoXWxCuHbM  
Added on Sep 11, 2023  
Last used within the last week — Read/write

At the bottom, there is a link to 'generating SSH keys' and 'common SSH problems'.

I have registered three SSH keys because I use three machines.

The link at the bottom has instructions for generating SSH keys on your machine.



# Generating SSH Keys

---

<https://docs.github.com/en/enterprise-server@3.9/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

## Generating a new SSH key

You can generate a new SSH key on your local machine. After you generate the key, you can add the public key to your account on your GitHub Enterprise Server instance to enable authentication for Git operations over SSH.

If you are a site administrator for your GitHub Enterprise Server instance, you can use the same key to grant yourself administrative SSH access to the instance. For more information, see "[Accessing the administrative shell \(SSH\)](#)."

- ① Open Terminal.
- ② Paste the text below, substituting in your GitHub Enterprise Server email address.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Use your northeastern.edu email address.



Northeastern University

# Generating SSH Keys

---

<https://docs.github.com/en/enterprise-server@3.9/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

- ③ When you're prompted to "Enter a file in which to save the key", you can press **Enter** to accept the default file location. Please note that if you created SSH keys previously, ssh-keygen may ask you to rewrite another key, in which case we recommend creating a custom-named SSH key. To do so, type the default file location and replace `id_ssh_keyname` with your custom key name.

```
> Enter a file in which to save the key (/Users/YOU/.ssh/id_ALGORITHM): [Press enter]
```

- ④ At the prompt, type a secure passphrase. For more information, see "[Working with SSH key passphrases](#)."

```
> Enter passphrase (empty for no passphrase): [Type a passphrase]
> Enter same passphrase again: [Type passphrase again]
```

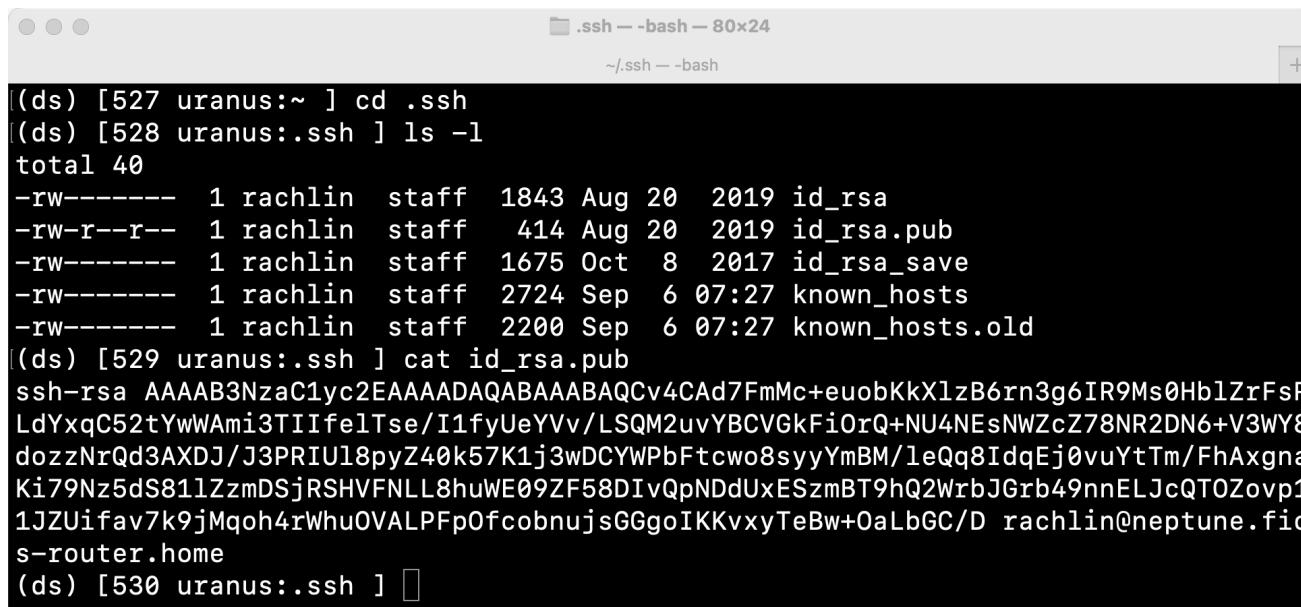
You can leave the passphrase empty for no passphrase.



# Your public and private ssh key

---

Your newly-generated public and private SSH key live in your home directory inside a .ssh folder.  
(The dot before the folder name indicates that this is a hidden folder.). I don't mind showing you my  
public key (**id\_rsa.pub**) but my private key (**id\_rsa**) is for my eyes only! Your keys might be named  
something different. Just look for the two files, one of which has the **.pub** extension. THAT's your public key!



The screenshot shows a terminal window titled ".ssh -- bash - 80x24" with the command "ls -l" run in the directory "~/.ssh". The output lists several files:

```
(ds) [527 uranus:~] cd .ssh
(ds) [528 uranus:.ssh] ls -l
total 40
-rw----- 1 rachlin staff 1843 Aug 20 2019 id_rsa
-rw-r--r-- 1 rachlin staff 414 Aug 20 2019 id_rsa.pub
-rw----- 1 rachlin staff 1675 Oct  8 2017 id_rsa_save
-rw----- 1 rachlin staff 2724 Sep  6 07:27 known_hosts
-rw----- 1 rachlin staff 2200 Sep  6 07:27 known_hosts.old
(ds) [529 uranus:.ssh] cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCV4CAD7FmMc+euobKkX1zB6rn3g6IR9Ms0HblZrFsR
LdYxqC52tYwWAmi3TIIfelTse/I1fyUeYVv/LSQM2uvYBCVGkFiOrQ+NU4NEsNWZcZ78NR2DN6+V3WY8
dozzNrQd3AXDJ/J3PRIU18pyZ40k57K1j3wDCYWPbFtcwo8syyYmBM/leQq8IdqEj0vuYtTm/FhAxgna
Ki79Nz5dS811ZzmDSjRSHVFNLL8huWE09ZF58DIVQpNDdUxESzmBT9hQ2WrbJGrb49nnELJcQT0Zovp1
1JZUifav7k9jMqoh4rWhu0VALPFp0fcobnujsGGgoIKKvxyTeBw+0aLbGC/D rachlin@neptune.fios-router.home
(ds) [530 uranus:.ssh] 
```



# Registering the public SSH key

## SSH keys / Add new

Title

Uranus Laptop

Key type

Authentication Key

Key

ssh-rsa

```
AAAAAB3NzaC1yc2EAAAQABAAQCV4CAd7FmMc+euobKkXlzB6rn3g6IR9Ms0HblZrFsRLdYxqC52tYwWAmi  
3TlIfeiTse/l1fyUeYVv/LSQM2uvYBCVGkFiOrQ+NU4NEsNWZcZ78NR2DN6+V3WY8dozzNrQd3AXDJ  
/J3PRIUI8pyZ40k57K1j3wDCYWPbFtcwo8syyYmBM/leQq8ldqEj0vuYtTm  
/FhAxgnaKi79Nz5dS81lZzmDSjRSHVFNLL8huWE09ZF58DlvQpNDdUxESzmBT9hQ2WrbJGrb49nnELJcQTOZovp1  
1JZUifav7k9jMqoh4rWhuOVALPFpOfcobnujsGGgoIKVxyTeBw+OaLbGC/D rachlin@neptune.fios-router.home
```

Add SSH key

Once the key is registered  
I can run git commands without  
having to enter a username  
and password.



Northeastern University

# Let's clone our test repo

Pull down the green “Code” button, select SSH, and copy that address to the clipboard.

The screenshot shows a GitHub repository page for 'rachlin / ds3500\_TEMP'. The 'Code' tab is selected. A dropdown menu is open under the 'Code' button, showing options: 'Clone' (with 'HTTPS', 'SSH', and 'GitHub CLI' tabs), a note about password-protected SSH keys, and links for 'Open with GitHub Desktop' and 'Download ZIP'. The 'SSH' tab is highlighted and circled in red. The URL 'git@github.khoury.northeastern.edu:rachli' is visible under the 'Clone' section.



Northeastern University

# Now clone the repo

---

```
(ds) [542 uranus:~] cd Documents
(ds) [543 uranus:Documents] cd courses
(ds) [544 uranus:courses] git clone git@github.khoury.northeastern.edu:rachlin/ds3500_TEMP.git
Cloning into 'ds3500_TEMP'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
Receiving objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
```



Pasted the repo address  
after the “git clone” command.



Northeastern University

# Adding a file to the repo

We create a subfolder and and a file called **test.dat** to the folder.

Next, we **add**, **commit**, and **push** the file in order to update the remote copy of the repo.

The Pro GIT book explains these commands in more detail. Please have a look.

```
(ds) [549 uranus:courses ] cd ds3500_TEMP/
(ds) [550 uranus:ds3500_TEMP ] ls
README.md
(ds) [551 uranus:ds3500_TEMP ] mkdir somefolder
(ds) [552 uranus:ds3500_TEMP ] cd somefolder
(ds) [553 uranus:somefolder ] echo "hello" > test.dat
(ds) [554 uranus:somefolder ] cat test.dat
hello
(ds) [555 uranus:somefolder ] git add test.dat
(ds) [556 uranus:somefolder ] git commit -m "Adding test.dat to the repo"
[main 3b7c09a] Adding test.dat to the repo
 1 file changed, 1 insertion(+)
   create mode 100644 somefolder/test.dat
(ds) [557 uranus:somefolder ] git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 382 bytes | 382.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.khoury.northeastern.edu:rachlin/ds3500_TEMP.git
  aeddc68..3b7c09a main -> main
(ds) [558 uranus:somefolder ] █
```



# Refresh our repo page on the git server

A screenshot of a GitHub repository page for 'rachlin / ds3500\_TEMP'. The repository is private. The 'Code' tab is selected. The main content area shows a list of commits:

- rachlin Adding test.dat to the repo (3b7c09a, 3 minutes ago, 2 commits)
- somefolder Adding test.dat to the repo (3 minutes ago)
- .gitignore Initial commit (30 minutes ago)
- README.md Initial commit (30 minutes ago)

The commit for 'somefolder' is circled in red. The 'About' section states: "This is a temporary repo, just for demonstration purposes". The 'Releases' section indicates "No releases published" and "Create a new release".



Northeastern University

# Obligatory XKCD



COMMENT	DATE
CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
ENABLED CONFIG FILE PARSING	9 HOURS AGO
MISC BUGFIXES	5 HOURS AGO
CODE ADDITIONS/EDITS	4 HOURS AGO
MORE CODE	4 HOURS AGO
HERE HAVE CODE	4 HOURS AGO
AAAAAAA	3 HOURS AGO
ADKFJSLKDFJSOKLFJ	3 HOURS AGO
MY HANDS ARE TYPING WORDS	2 HOURS AGO
HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.



# Reliability, Scalability, and Maintainability

---



Northeastern University

# Challenges

---

- **Reliability:** The system should continue to work *correctly* in response to hardware, software, or user errors
- **Scalability:** As the system grows (data volume, traffic, or complexity) there should be reasonable ways of dealing with that growth. The design should be *elastic*.
- **Maintainability:** Productively work on the system as people come and go. Add new features to adapt to changing requirements.



# What is reliability?

---

- **Predictable:** Application performs as expected
- **Fault-tolerant:** Application copes with human error & hardware failures.
- **Performant:** Performance is adequate under expected load and data volume
- **Secure:** System prevents unauthorized access



# Reliability

- **Faults** are a deviation in the spec of one particular component of the system. **Failures** cause the whole system to stop, requiring user intervention.
- A system is *fault-tolerant* if it anticipates and copes with certain kind of faults.
- One way to design fault-tolerant systems is to induce them deliberately! (e.g., submit bad input by the user, shut down a process, etc.)



Northeastern University

Chaos Monkey  Enabled

Termination frequency

Mean time between terms  days ⓘ      Minimum time between terms  days ⓘ

Grouping ⓘ  App  Stack  Cluster  Regions are independent ⓘ

Exceptions ⓘ

Account	Region	Stack	Detail	Remove
prod	us-west-2	staging		
test	*	*	*	

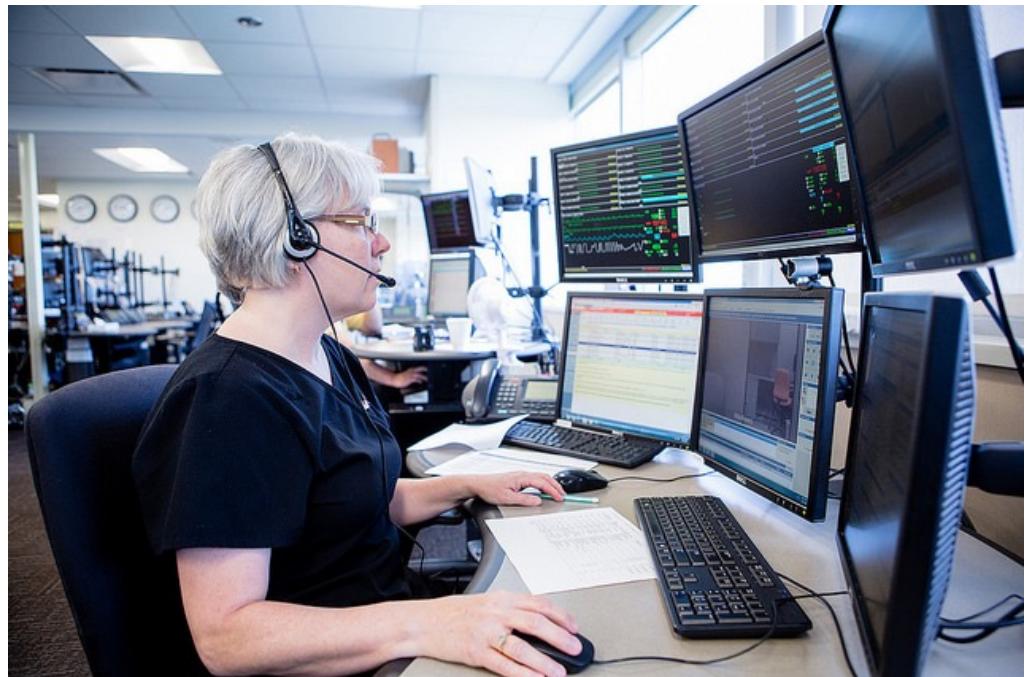
[Add Exception](#)

Netflix Chaos Monkey

# Reliability: Human Errors

---

- Application / service / network configuration errors
- Balance flexibility with restrictions aimed at encouraging users to do the right thing
- Sandbox testing/ Automated unit testing to identify edge cases
- Automated rollout / rollback of code changes (DevOps)
- Monitoring / Dashboards / Telemetry for error detection, performance tracking



# Maintainability

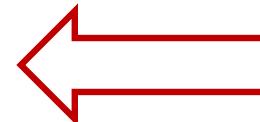
---

## Software cost breakdown:

Initial development: ~10-20%

Error correction: ~10-20%

On-going Maintenance: ~60-80%



**Operability:** Make it easy for operations teams to keep system running

**Simplicity:** Make it easy for engineers to understand

**Evolvability:** Make it easy for engineers to modify in response to changing requirements (*extensibility / plasticity*)



# Python Exception Hierarchy

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
        +-- FloatingPointError
        +-- OverflowError
        +-- ZeroDivisionError
    +-- AssertionError
    +-- AttributeError
    +-- BufferError
    +-- EOFError
    +-- ImportError
        +-- ModuleNotFoundError
    +-- LookupError
        +-- IndexError
        +-- KeyError
    +-- MemoryError
    +-- NameError
        +-- UnboundLocalError
    +-- OSError
        +-- BlockingIOError
        +-- ChildProcessError
        +-- ConnectionError
            +-- BrokenPipeError
            +-- ConnectionAbortedError
            +-- ConnectionRefusedError
            +-- ConnectionResetError
        +-- FileExistsError
        +-- FileNotFoundError
        +-- InterruptedError
        +-- IsADirectoryError
        +-- NotADirectoryError
        +-- PermissionError
        +-- ProcessLookupError
        +-- TimeoutError
```

```
+-- ReferenceError
+-- RuntimeError
|   +-- NotImplementedError
|   +-- RecursionError
+-- SyntaxError
|   +-- IndentationError
|       +-- TabError
+-- SystemError
+-- TypeError
+-- ValueError
|   +-- UnicodeError
|       +-- UnicodeDecodeError
|       +-- UnicodeEncodeError
|       +-- UnicodeTranslateError
+-- Warning
    +-- DeprecationWarning
    +-- PendingDeprecationWarning
    +-- RuntimeWarning
    +-- SyntaxWarning
    +-- UserWarning
    +-- FutureWarning
    +-- ImportWarning
    +-- UnicodeWarning
    +-- BytesWarning
    +-- ResourceWarning
```

In Python, all exceptions must be instances of a class that derives from [BaseException](#). In a `try` statement with an [`except`](#) clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived).

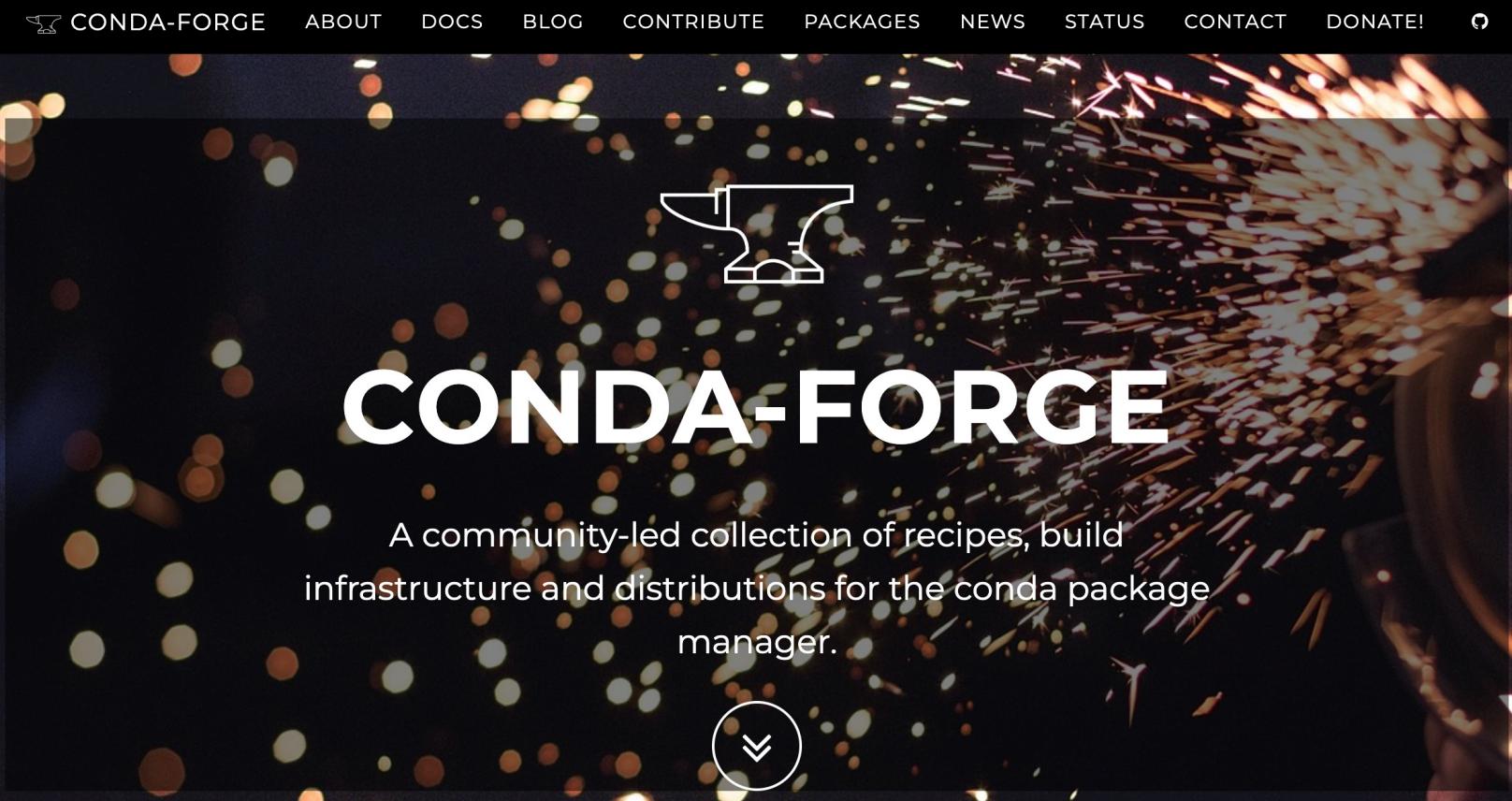
# Advanced Visualization with Plotly



Northeastern University

# Conda Forge (conda-forge.org)

---



The image shows the Conda Forge homepage. At the top, there is a navigation bar with links: CONDA-FORGE, ABOUT, DOCS, BLOG, CONTRIBUTE, PACKAGES, NEWS, STATUS, CONTACT, DONATE!, and a user icon. Below the navigation bar is a large banner featuring a dark background with a grid of glowing orange and yellow particles. In the center, there is a white icon of a hammer and anvil. Below the icon, the word "CONDA-FORGE" is written in large, bold, white capital letters. Underneath the main title, there is a description in white text: "A community-led collection of recipes, build infrastructure and distributions for the conda package manager." At the bottom of the banner, there is a circular button with a downward-pointing arrow.



Northeastern University

# Why use the conda-forge channel?

---

The conda team, from [Anaconda, Inc.](#), packages a multitude of packages and provides them to all users free of charge in their `default` channel.

But what if a package you are looking for is not in the default channel? In the past users only had the option to create an [Anaconda Cloud](#) account and create their own channel.

[conda-forge](#) is a GitHub organization containing repositories of conda recipes. Thanks to some awesome continuous integration providers (AppVeyor, Azure Pipelines, CircleCI and TravisCI), each repository, also known as a feedstock, automatically builds its own recipe in a clean and repeatable way on Windows, Linux and OSX.

```
conda config --add channels conda-forge  
conda config --set channel_priority strict  
conda install <package-name>
```



# Channel Priority

---

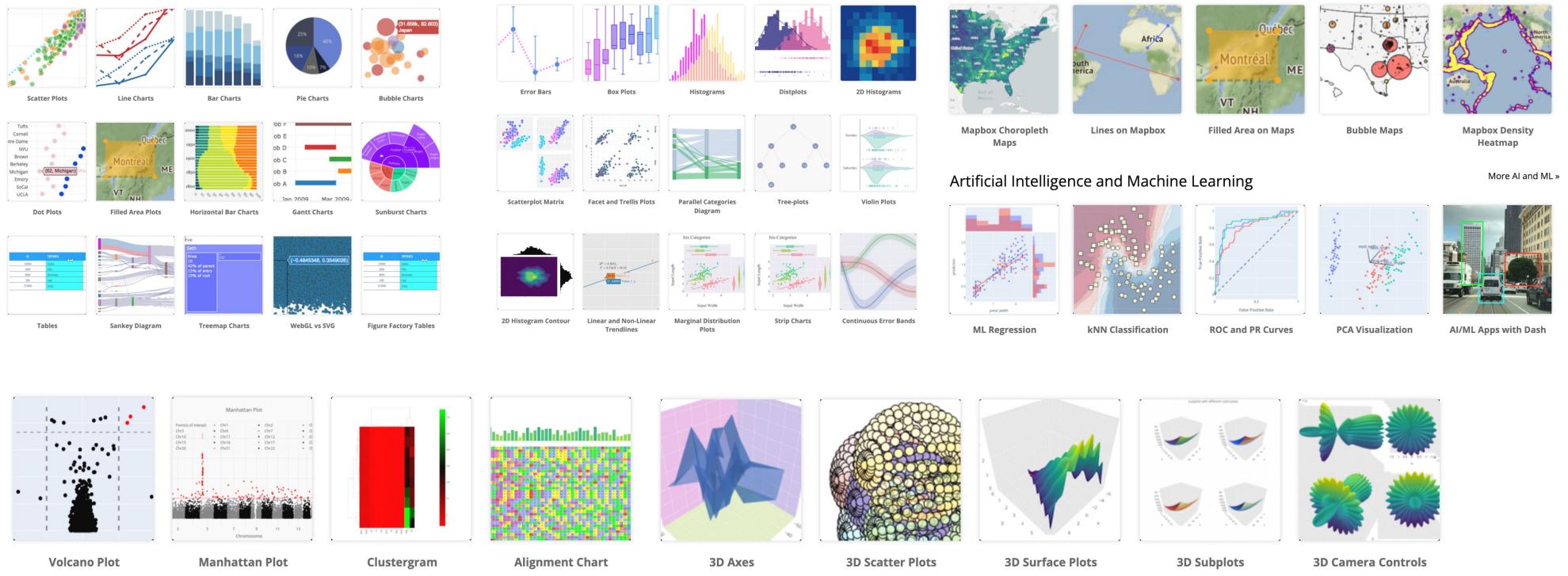
Documentation: **conda config --describe channel\_priority**

Setting the channel priority: **conda config --set channel\_priority <strict | flexible | disabled>**

- strict:** Packages in lower priority channels are not considered if a package with the same name appears in a higher priority channel.  
(Use only the first channel that has the package to resolve any dependencies.)
- flexible:** With flexible channel priority, the solver may reach into lower priority channels to fulfill dependencies, rather than raising an unsatisfiable error. (A higher priority channel with an earlier package version takes precedence over a lower priority channel with a newer package version.)
- disabled:** With channel priority disabled, package version takes precedence, and the configured priority of channels is used only to break ties.



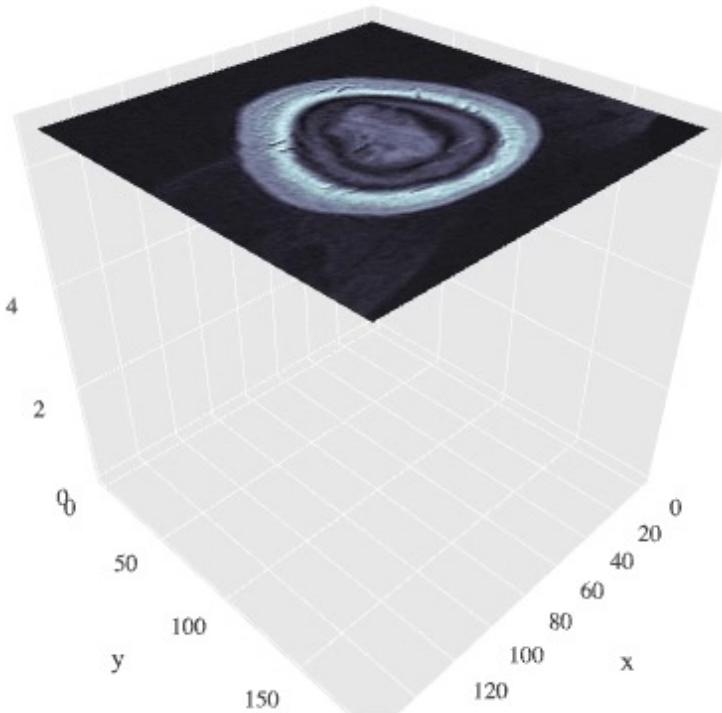
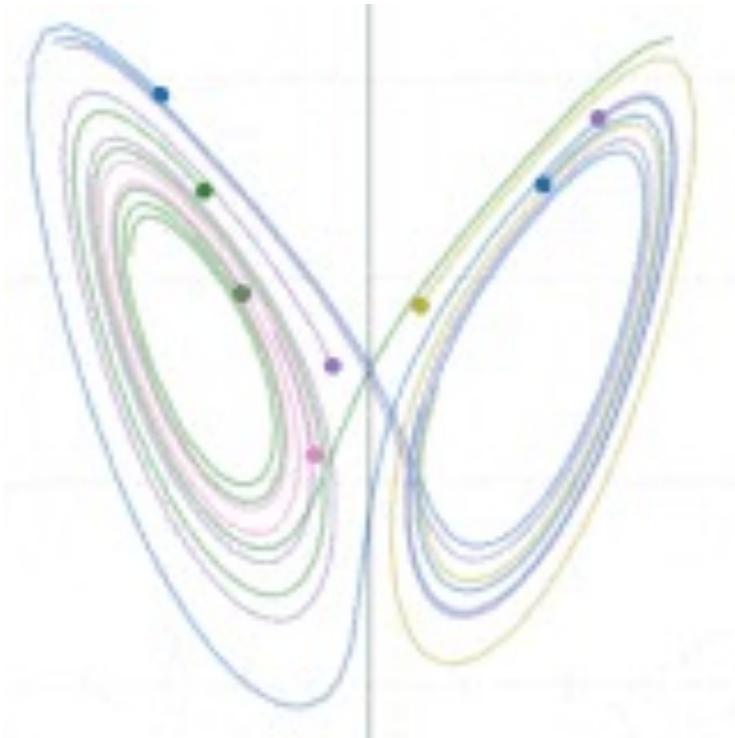
# A gallery of possibilities: <https://plotly.com/python/>



Northeastern University

# Animations!

---



## Is Plotly for Python Free?

**Yes.** Plotly for Python is free and open-source software, [licensed under the MIT license](#). It costs nothing to [install and use](#). You can view the source, report issues or contribute using [our Github repository](#).

## Can I use Plotly for Python without signing up to any service?

**Yes.** You can use Plotly for Python to make, view, and distribute charts and maps without registering for any service, obtaining any token, or creating any account. The one exception is that to view tile maps which use tiles from the Mapbox service (which is optional, as [you can use other tile servers](#)), you will need to have a Mapbox token.

## Can I use Plotly for Python offline, without being connected to the internet?

**Yes.** You can use Plotly for Python to make, view, and distribute graphics totally offline. The one exception is that to view tile maps which use tiles from a cloud-hosted service, such as Open Street Maps or Mapbox, you will need a connection to that service. You can view tile maps totally offline if you run your own local tile server and [use its tiles](#).



# Free Software: Free as in beer / Free as in speech

---

Free as in Beer



Free as in Speech



- The software doesn't cost you anything (monetarily) to use.
- There are still restrictions on **how** you use the software – for example you can't necessarily view, modify, or redistribute the source code.

- You are at liberty to use the software however you like.
- You can inspect the source code.
- You can redistribute or repackage the code.
- You can create your own improved versions.



Northeastern University

# Installation

---

```
$ conda install -c conda-forge plotly
```



```
The following packages will be downloaded:
package          |       build
-----|-----
plotly-5.12.0    | pyhd8ed1ab_1      5.7 MB  conda-forge
-----|-----
                                         Total: 5.7 MB

The following NEW packages will be INSTALLED:
plotly           conda-forge/noarch::plotly-5.12.0-pyhd8ed1ab_1 None
tenacity         conda-forge/noarch::tenacity-8.1.0-pyhd8ed1ab_0 None

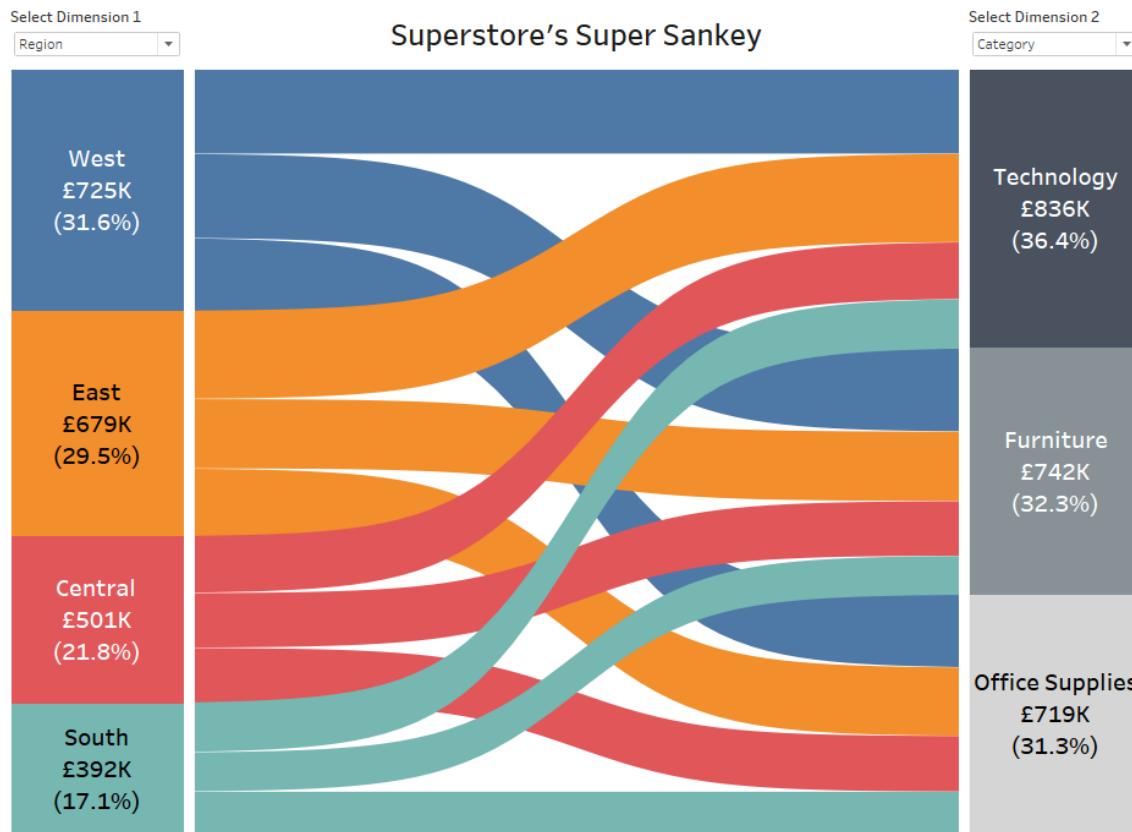
Proceed ([y]/n)?
```

Downloading and Extracting Packages
plotly-5.12.0 | 5.7 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done
(ds) [530 uranus:~ ]



Northeastern University

# Sankey Diagrams: Market Analysis

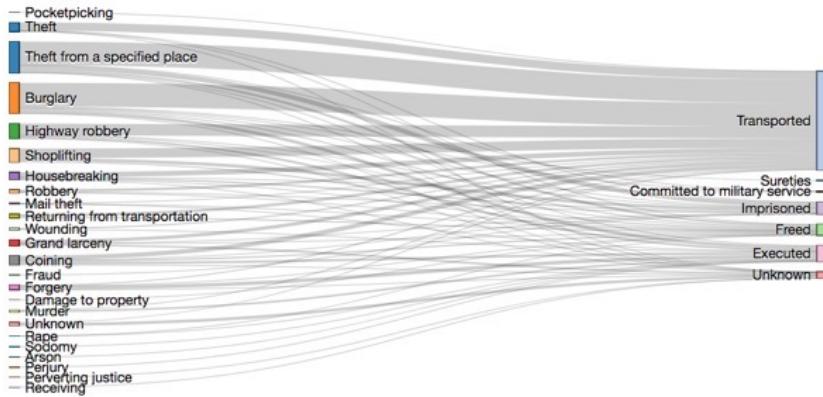


Northeastern University

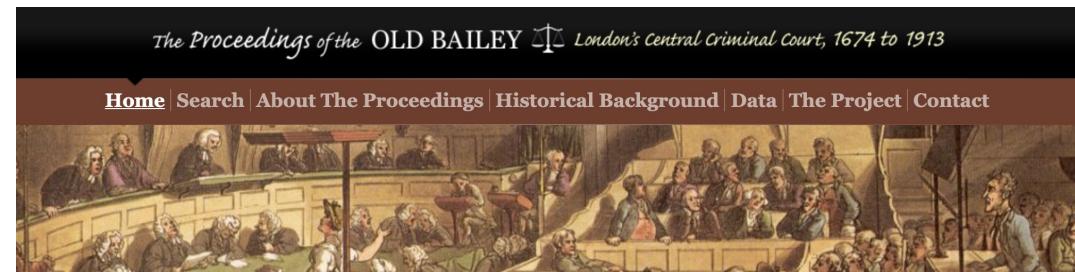
# Crime and Punishment in the Old Bailey

## What actually happened to defendants sentenced to death?

763 results [Change visualisation](#) [View as list](#)



Offence is shown on the left and sentence outcome on the right. All of these defendants were sentenced to death between 1810 and 1815. But many Old Bailey defendants who were sentenced to death were not actually executed. This Sankey diagram shows what actually happened to them. [See all the results](#)



[Home Page](#)

[Search](#)

[About the Proceedings](#)

[Historical Background](#)

[API](#)

[The Project](#)

[Copyright & Citation Guide](#)

[Contact](#)

ON THIS DAY IN... **1722**

James Lanman, (aged 11) and Samuel Armstrong, (aged 13) allegedly stole a silver snuffbox from Simon Hansel's shop. A month later they were sentenced to hang. [read more](#)

## The Proceedings of the Old Bailey, 1674-1913

A fully searchable edition of the largest body of texts detailing the lives of non-elite people ever published, containing 197,745 criminal trials held at London's central criminal court. If you are new to this site, you may find the [Getting Started](#) and [Guide to Searching](#) videos and tutorials helpful.

To search the Proceedings use the boxes on the right or go to the [Search Pages](#).

This site uses cookies. See our [privacy policy](#).

### February 2018 Update

Several bugs have been fixed and tagging errors corrected. In addition, the following new features have been added:

SEARCH  
the Proceedings

Keyword(s)

Reference No.

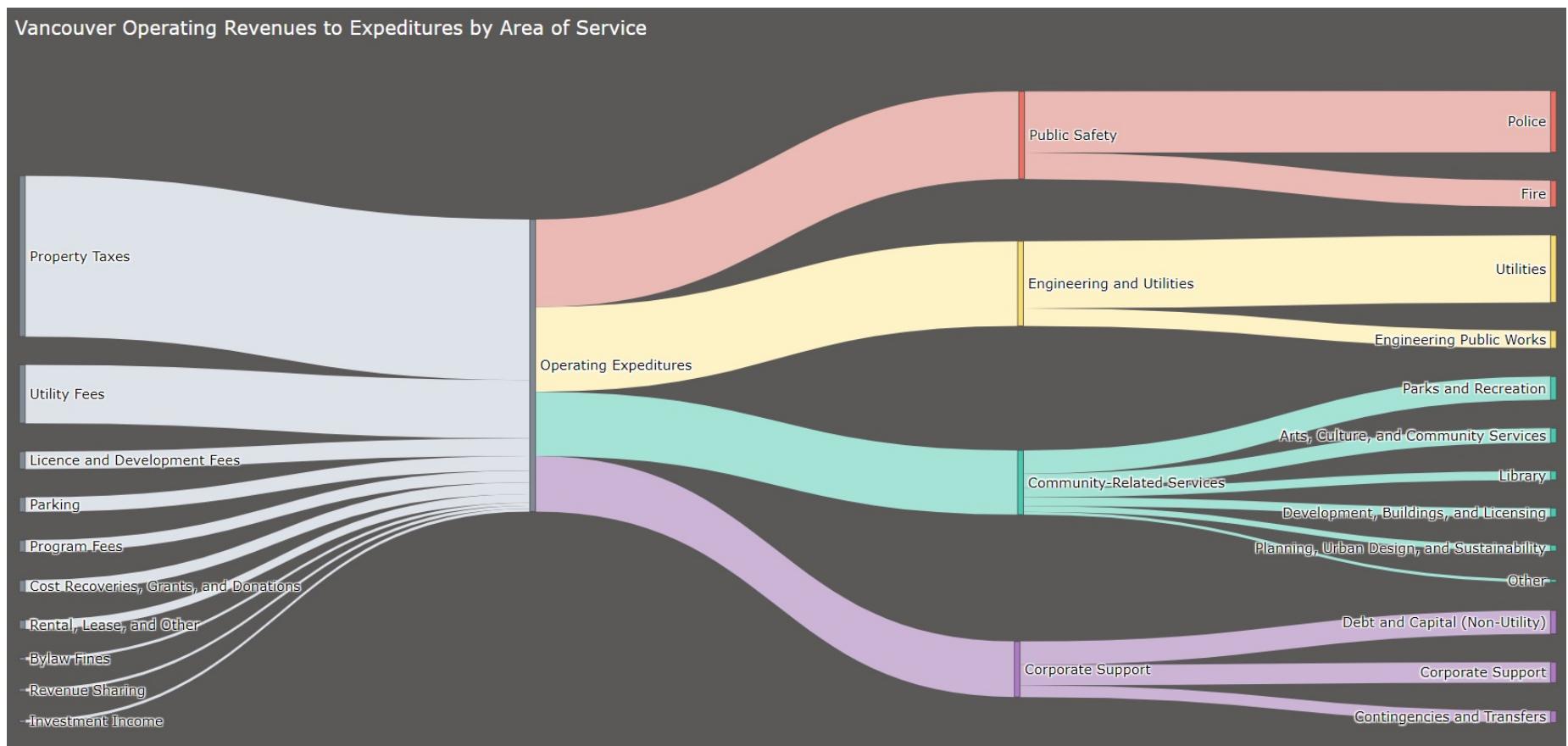
Search In

The Proceedings can also be searched in:



Northeastern University

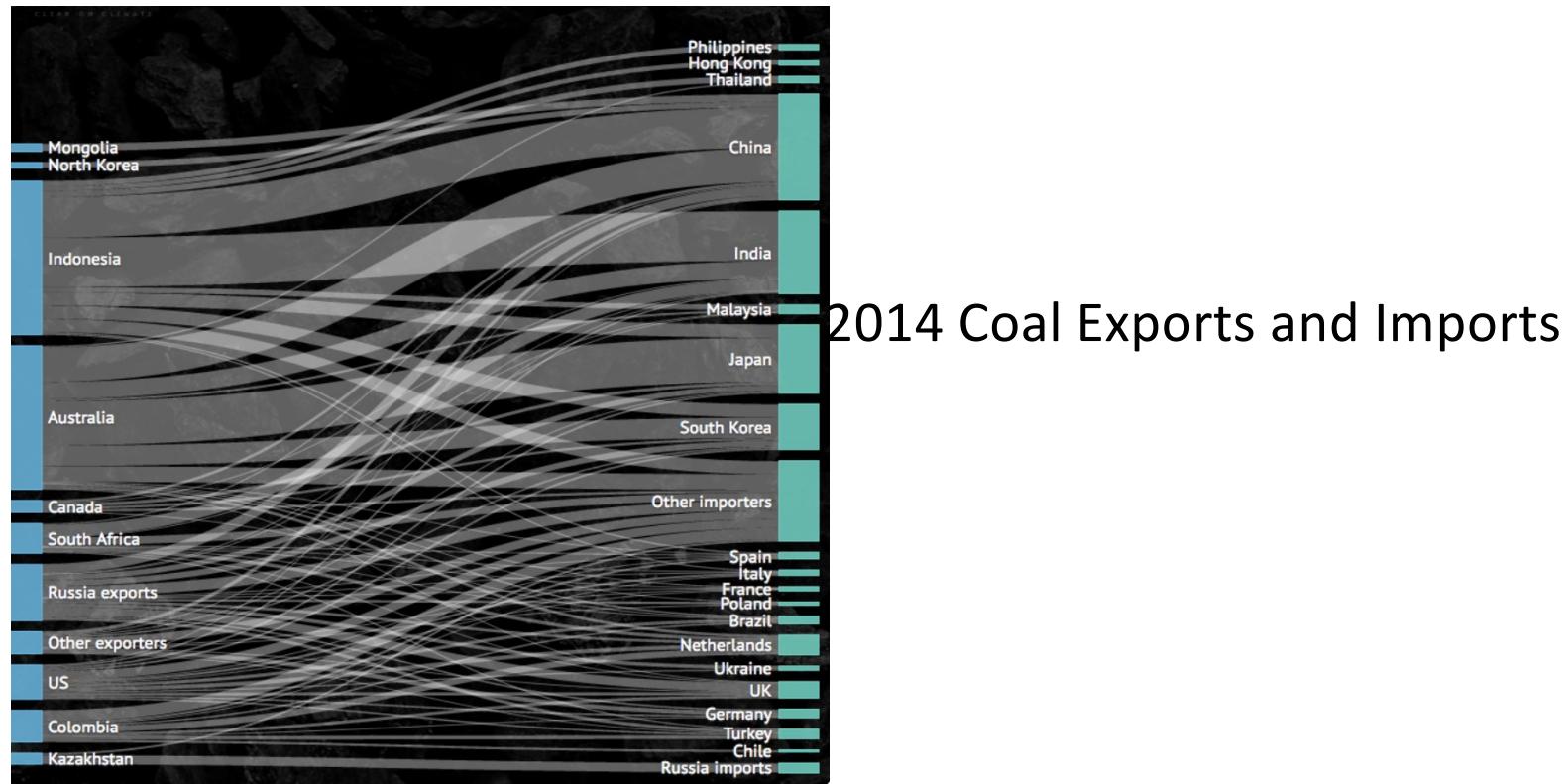
# Sankey Diagrams: Analyzing Expenses



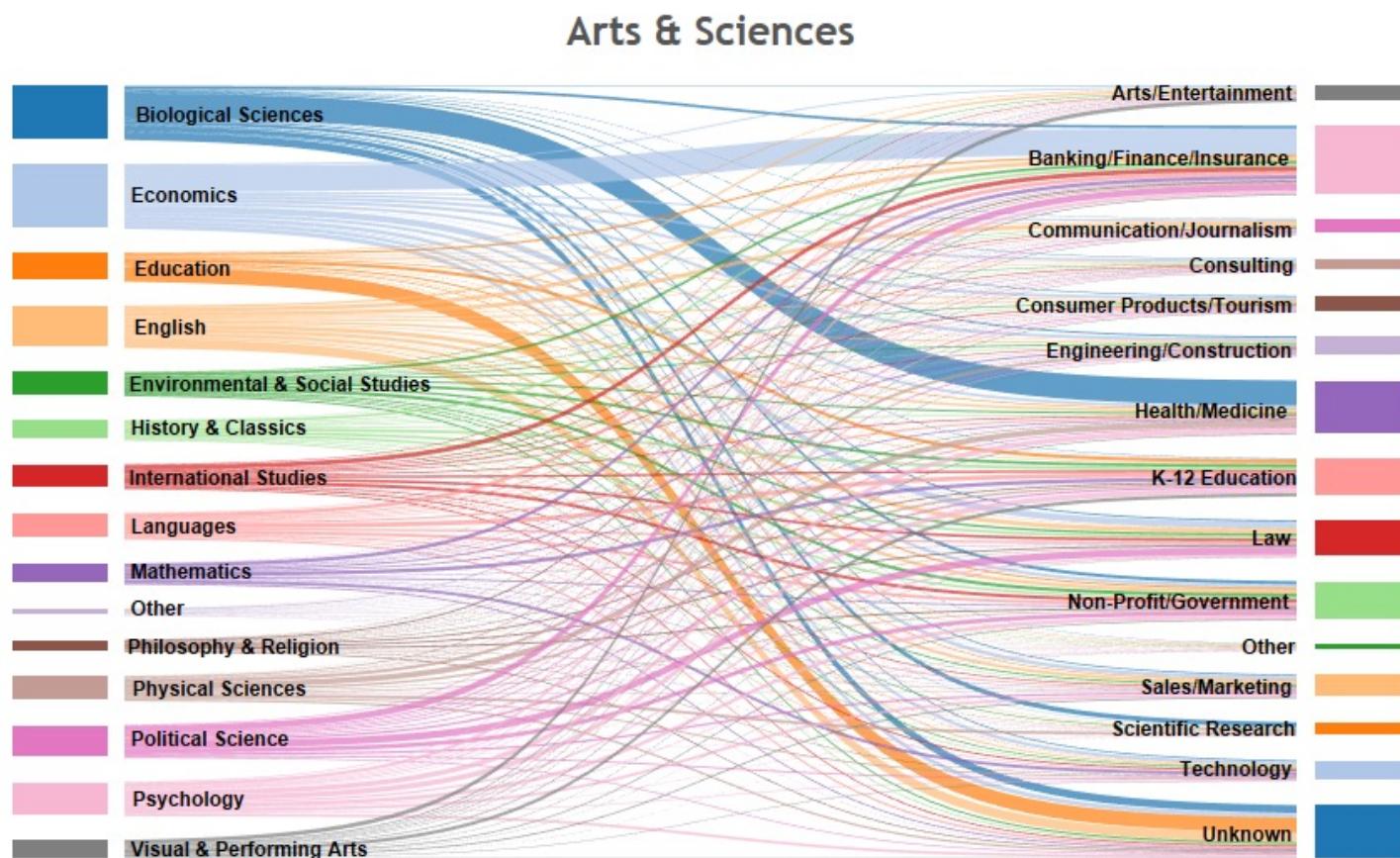
Northeastern University

# Sankey Diagrams: Trade

---

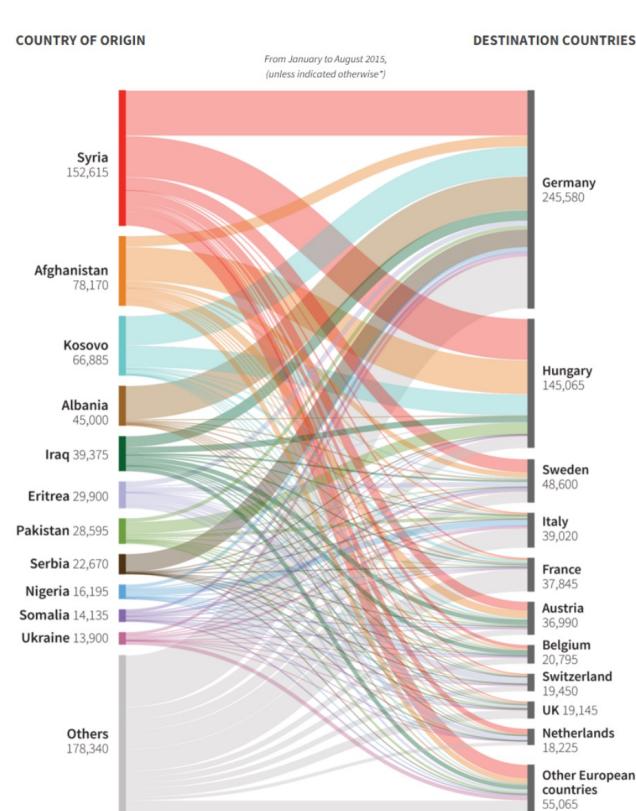


# Sankey Diagrams: The value of a liberal arts education



# Sankey Diagrams: Migration Patterns

Asylum seekers in Europe



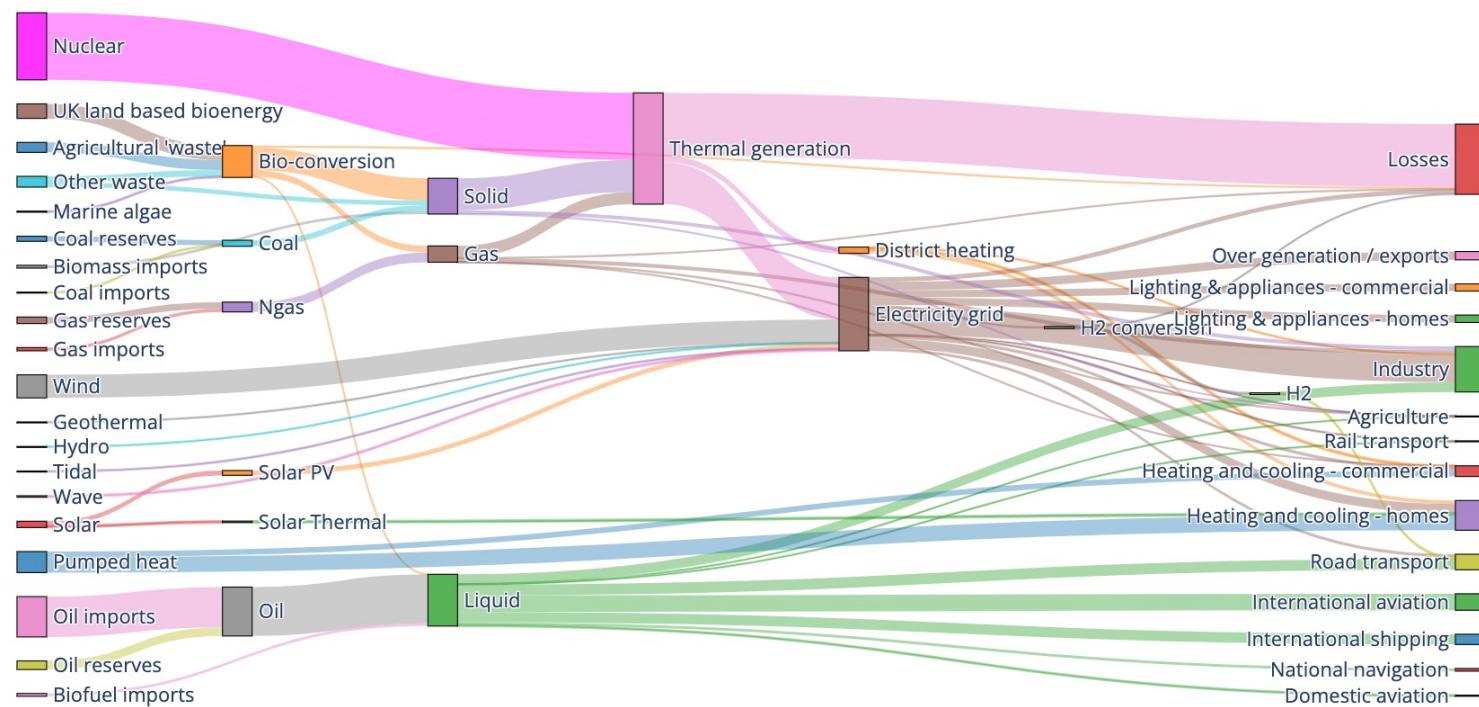
\*Latest data available for Czech Republic, Denmark, Estonia, Ireland, Greece, France, Italy, Latvia, Malta, Austria, Portugal, Romania, Slovakia, Finland, UK, and Iceland is for July. Latest data available for Spain, Cyprus and Liechtenstein is for the month of June. Eurostat data as of Oct. 2.



# Energy Forecasting

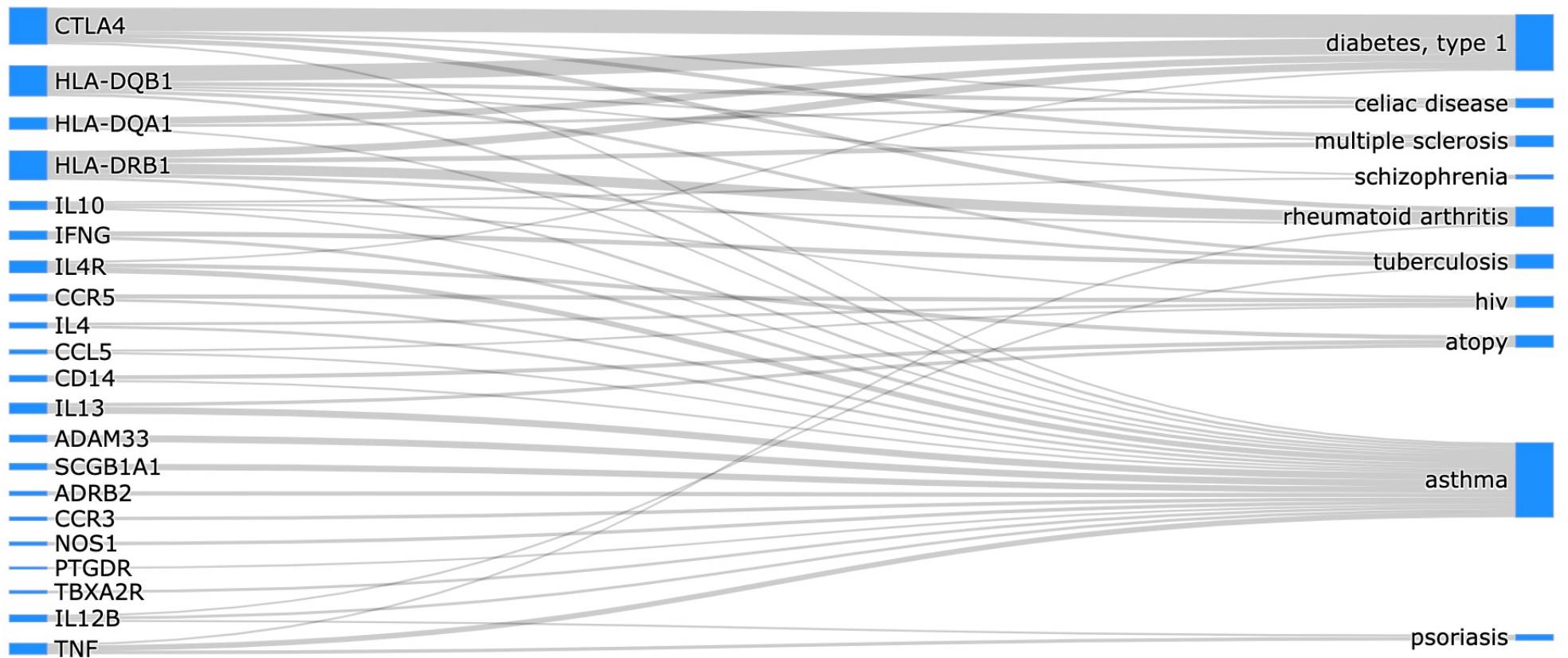
Energy forecast for 2050

Source: Department of Energy & Climate Change, Tom Counsell via [Mike Bostock](#)



Northeastern University

# Sankey: The Genetic Association Database



# Links between Asthma and Type 1 Diabetes

This Issue Views 3,824 | Citations 5 | Altmetric 40 | Comments 1

PDF



More ▾



Cite



Permissions

Original Investigation | Pediatrics



March 12, 2020

## Familial Coaggregation of Asthma and Type 1 Diabetes in Children

Awad I. Smew, MD<sup>1</sup>; Cecilia Lundholm, MSc<sup>1</sup>; Lars Sävendahl, MD, PhD<sup>2</sup>; et al

» Author Affiliations | Article Information

JAMA Netw Open. 2020;3(3):e200834. doi:10.1001/jamanetworkopen.2020.0834

### Key Points | Español | 中文 (Chinese)

**Question** Is there an association between childhood asthma and type 1 diabetes and do shared familial factors contribute to the comorbidity?



Northeastern University

Wolters Kluwer

Medicine®

Home

Search

Submit a Manuscript

Medicine (Baltimore). 2015 Sep; 94(36): e1466.

Published online 2015 Sep 11.

doi: 10.1097/MD.0000000000001466

PMCID: PMC46166

PMID: 263567

### Type 1 Diabetes and Increased Risk of Subsequent Asthma

A Nationwide Population-Based Cohort Study

Yung-Tsung Hsiao, MD, Wen-Chien Cheng, MD, Wei-Chih Liao, MD, Cheng-Li Lin, MS, Te-Chun Shen, MD, Wei-Chun Chen, MD, Chia-Hung Chen, MD, and Chia-Hung Kao, MD

Monitoring Editor: Zhu. Xiaolin

» Author information » Article notes » Copyright and License information | Disclaimer

This article has been cited by other articles in PMC.

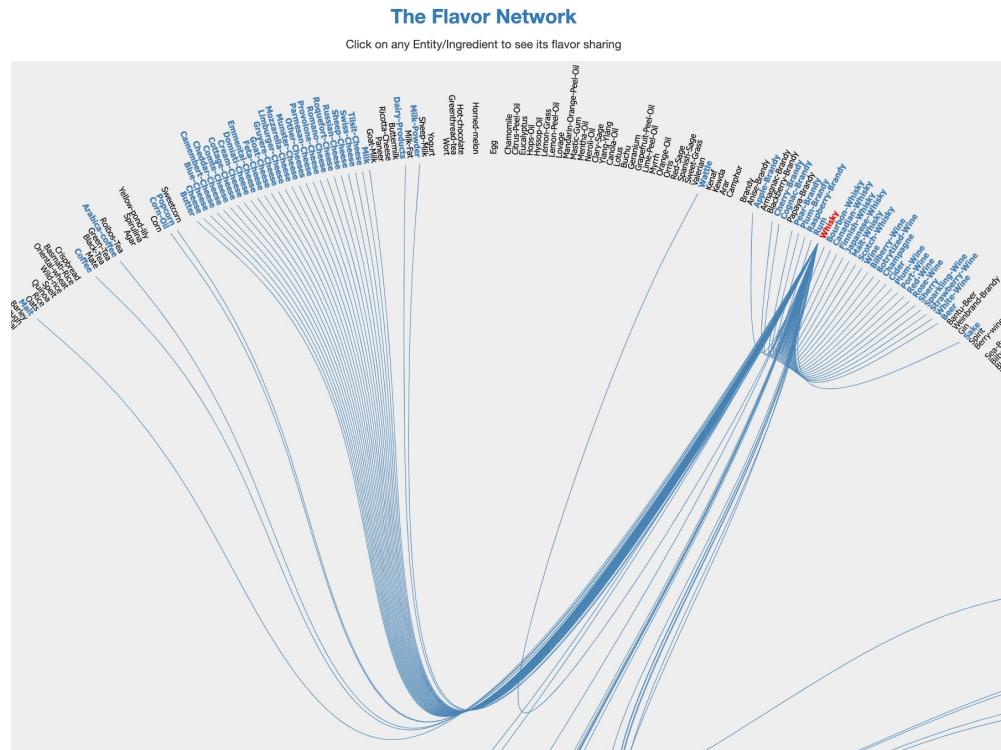
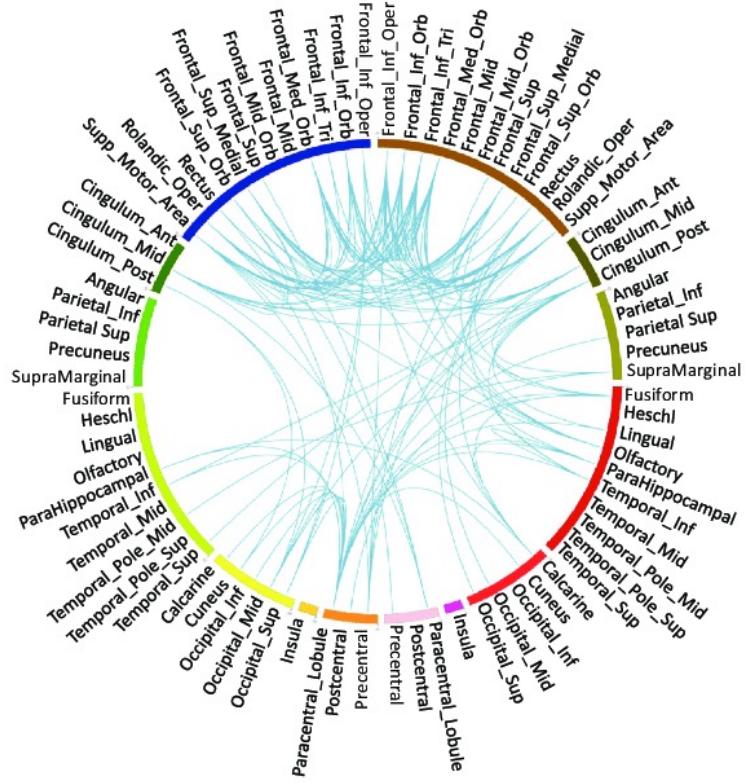
### Abstract

Search | Copy

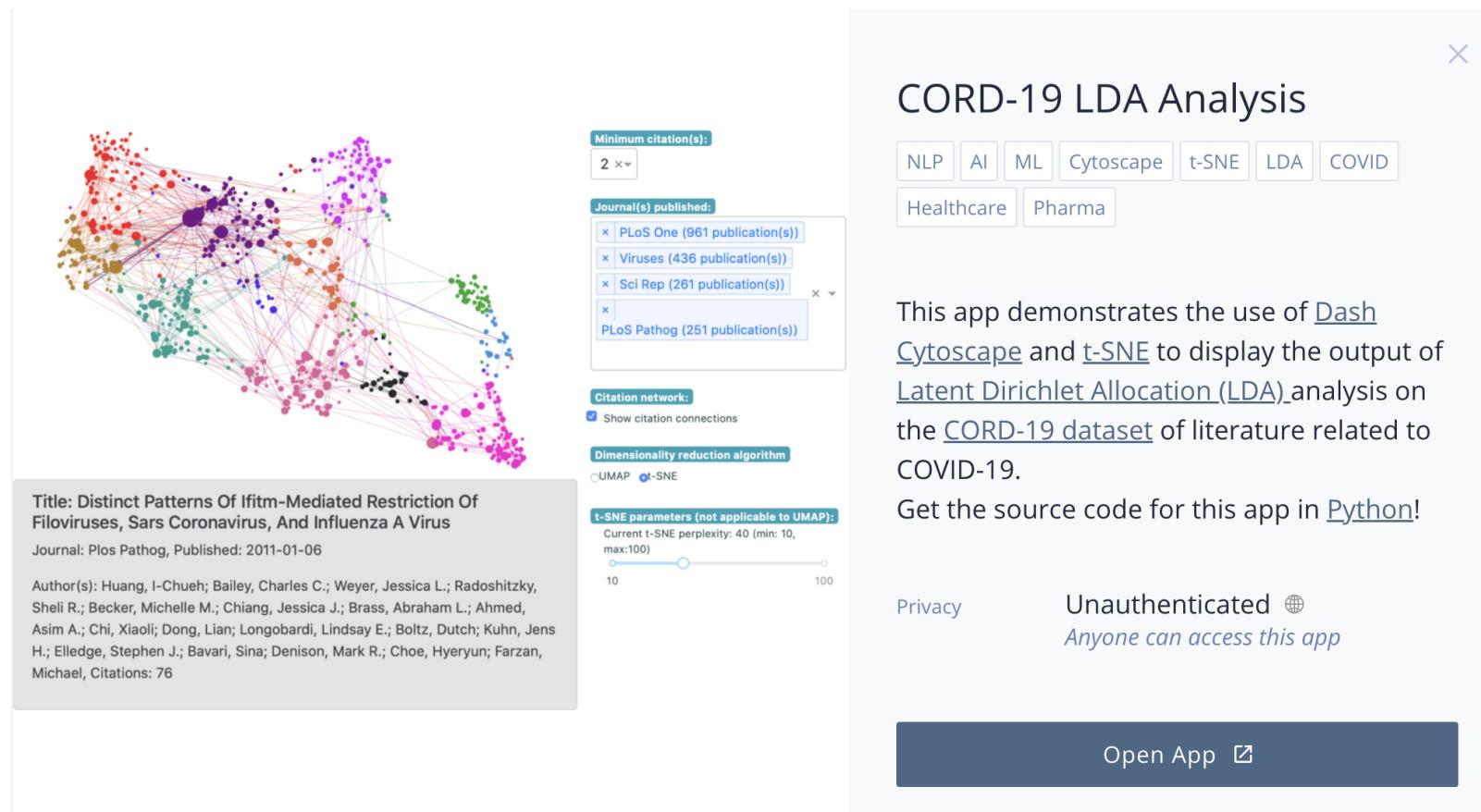
Go to:

The association between type 1 diabetes mellitus (T1DM) and asthma remains controversial and has led to new interest in these 2 disorders. The purpose of this study was to examine the associations among young people with T1DM and asthma and offer a clinical demonstration of the balance between Th1 and Th2 responses.

# Circle Graphs: Single-Set Sankey



# Dashboarding for Interactive Visualization

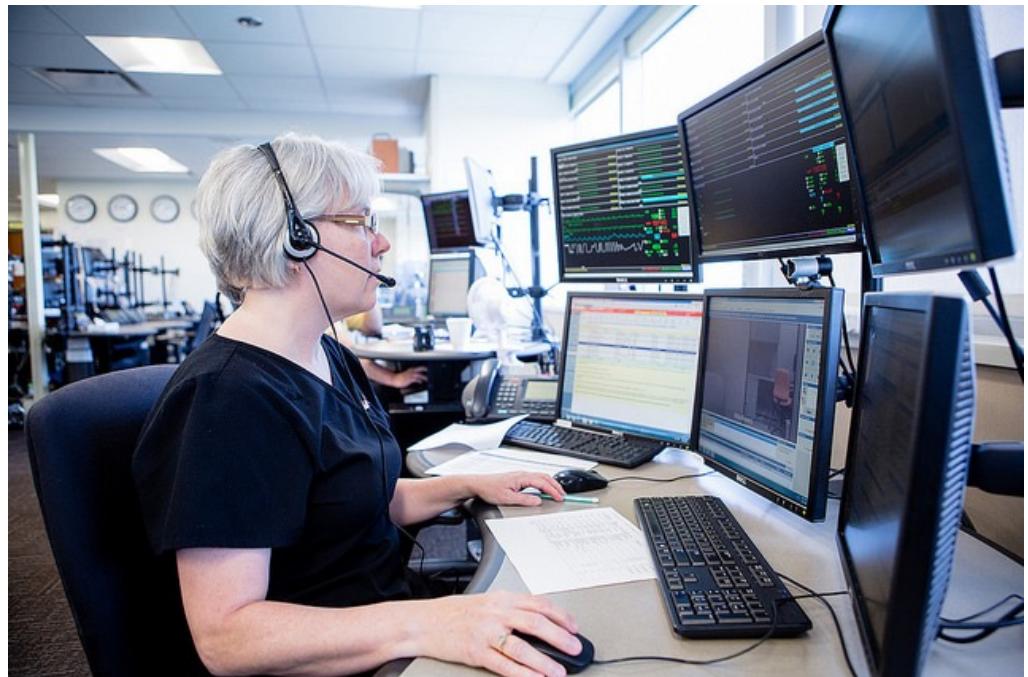


Northeastern University

# Monitoring Data Pipelines

---

- Application / service / network configuration errors
- Balance flexibility with restrictions aimed at encouraging users to do the right thing
- Sandbox testing/ Automated unit testing to identify edge cases
- Automated rollout / rollback of code changes (DevOps)
- Monitoring / Dashboards / Telemetry for error detection, performance tracking



# Dashboards – cont.



Northeastern University

```
$ conda install dash
```

```
The following packages will be downloaded:
```

package	build		
brotli-python-1.0.9	py39hfd1d529_7	736 KB	conda-forge
dash-2.3.1	pyhd8ed1ab_0	7.5 MB	conda-forge
flask-compress-1.11	pyhd8ed1ab_0	14 KB	conda-forge
Total:			8.3 MB

```
The following NEW packages will be INSTALLED:
```

```
brotli-python      conda-forge/osx-64::brotli-python-1.0.9-py39hfd1d529_7
dash              conda-forge/noarch::dash-2.3.1-pyhd8ed1ab_0
flask-compress    conda-forge/noarch::flask-compress-1.11-pyhd8ed1ab_0
```



# \$ conda install jupyter-dash

---

If you prefer using Jupyter Notebooks or Jupyter Lab

```
The following packages will be downloaded:
```

package	build		
ansi2html-1.7.0	py39h6e9494a_1	39 KB	conda-forge
jupyter-dash-0.4.2	pyhd8ed1ab_1	20 KB	conda-forge
retrying-1.3.3	py_2	11 KB	conda-forge

```
Total: 71 KB
```

```
The following NEW packages will be INSTALLED:
```

ansi2html	conda-forge/osx-64::ansi2html-1.7.0-py39h6e9494a_1
jupyter-dash	conda-forge/noarch::jupyter-dash-0.4.2-pyhd8ed1ab_1
retrying	conda-forge/noarch::retrying-1.3.3-py_2



# Dash installation verification: \$ conda list dash

---

```
(base) [512 uranus:dash ] conda list dash
# packages in environment at /Users/rachlin/opt/anaconda3:
#
#           Name          Version      Build  Channel
dash            2.3.1       pyhd8ed1ab_0  conda-forge
jupyter-dash    0.4.2       pyhd8ed1ab_1  conda-forge
```

**conda list plotly** (plotly and plotly express already installed)

```
(base) [513 uranus:dash ] condalist plotly
plotly           5.5.0       pyhd8ed1ab_0  conda-forge
plotly_express   0.4.1           py_0        conda-forge
```



# Dash HTML Components

```
from dash import html

html.Div([
    html.H1('Hello Dash'),
    html.Div([
        html.P('Dash converts Python classes into HTML'),
        html.P("This conversion happens behind the scenes by Dash's JavaScript front-end")
    ])
])
```

which gets converted (behind the scenes) into the following HTML in your web-app:

```
<div>
    <h1>Hello Dash</h1>
    <div>
        <p>Dash converts Python classes into HTML</p>
        <p>This conversion happens behind the scenes by Dash's JavaScript front-end</p>
    </div>
</div>
```



Northeastern University

Source: <https://dash.plotly.com/dash-html-components>

# Dash Core Components

---

## Dropdown

```
from dash import Dash, html, dcc

app = Dash(__name__)

app.layout = html.Div([
    dcc.Dropdown(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



Montréal

A screenshot of a dropdown menu interface. The word "Montréal" is displayed in a white box, indicating it is the selected item. There are arrows at the top right of the box and a close button (an 'x') to its right.

# Dash Core Components → localhost:8050

## Slider

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Slider(-5, 10, 1, value=-3)
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



# Dash Core Components

---

## Radio Items

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.RadioItems(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



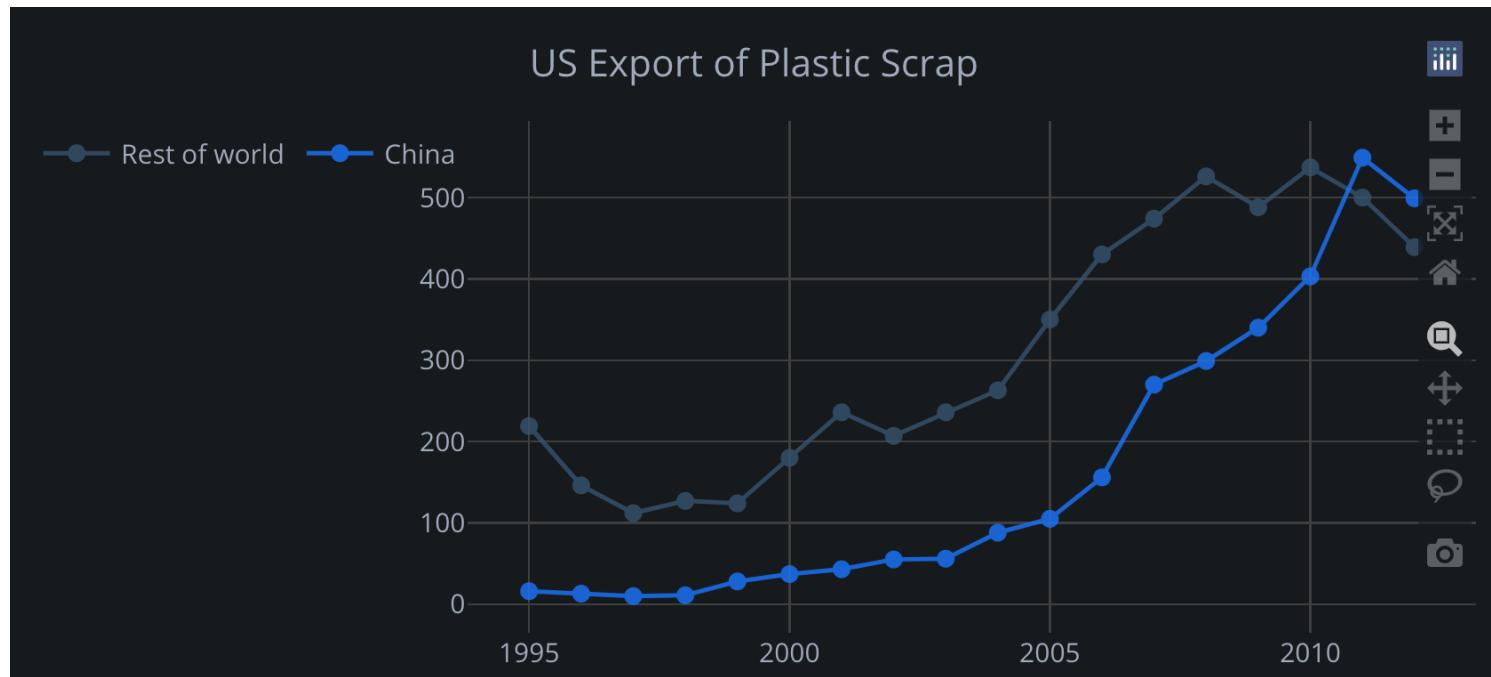
- New York City
- Montréal
- San Francisco



# Graphs! (Including our friend, the SanKey Diagram)

## Graphs

The `Graph` component shares the same syntax as the open-source `plotly.py` library. View the [plotly.py docs](#) to learn more.



# Live Updates

---



## Live Updating Components

### The `dcc.Interval` component

Components in Dash usually update through user interaction: selecting a dropdown, dragging a slider, hovering over points.

If you're building an application for monitoring, you may want to update components in your application every few seconds or minutes.

The `dcc.Interval` element allows you to update components on a predefined interval. The `n_intervals` property is an integer that is automatically incremented every time `interval` milliseconds pass. You can listen to this variable inside your app's `callback` to fire the callback on a predefined interval.



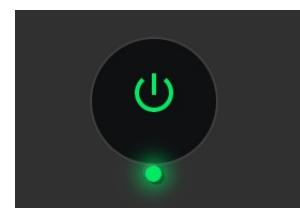
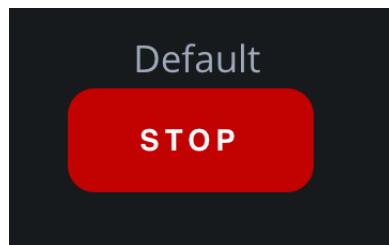
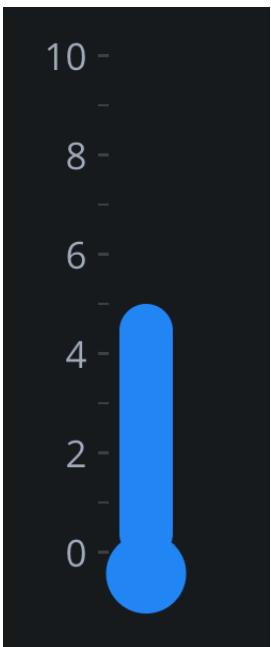
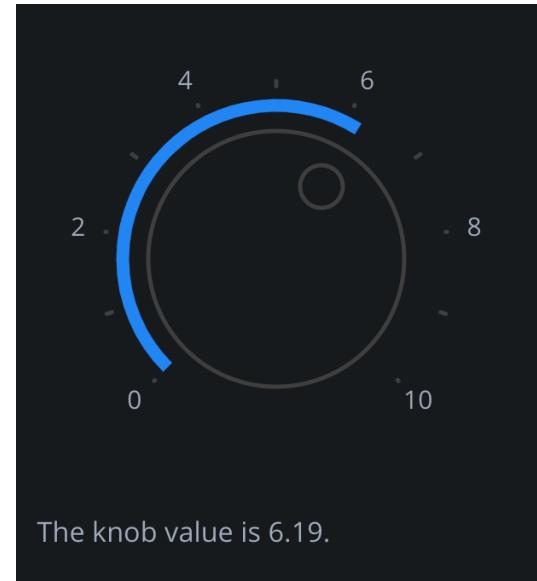
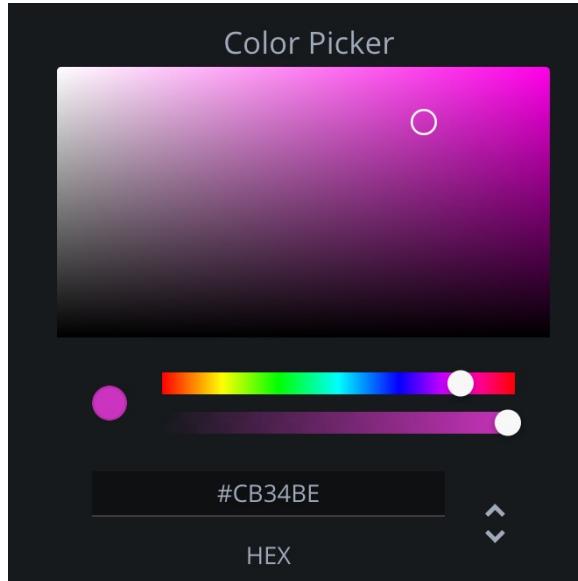
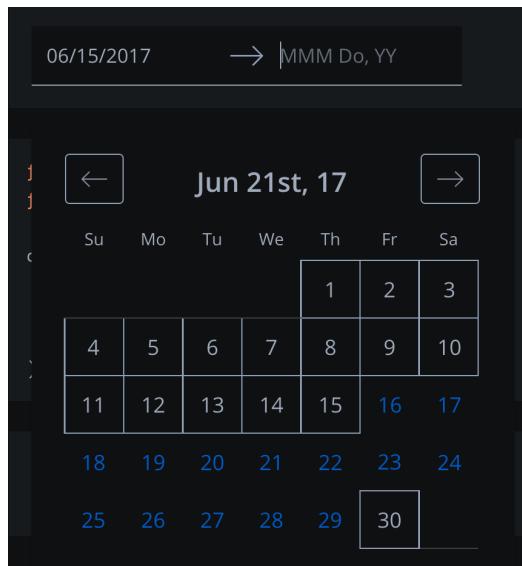
# Dash Tables

♦	index	♦	continent	♦	country	♦	gdpPercap	♦	lifeExp	♦	pop
	1		Asia		Afghanistan		974.5803384		43.828		31889923
	2		Europe		Albania		5937.029525999999		76.423		3600523
	3		Africa		Algeria		6223.367465		72.301		33333216
	4		Africa		Angola		4797.231267		42.731		12420476
	5		Americas		Argentina		12779.37964		75.32		40301927

« < 1 > »



# Dash DAQ: Web development framework



Northeastern University

# Callbacks

Callbacks make your dashboards interactive

```
from dash import Dash, dcc, html, Input, Output

app = Dash(__name__)
app.layout = html.Div([
    dcc.Dropdown(['NYC', 'MTL', 'SF'], 'NYC', id='demo-dropdown'),
    html.Div(id='dd-output-container')
])

@app.callback(
    Output('dd-output-container', 'children'),
    Input('demo-dropdown', 'value')
)
def update_output(value):
    return f'You have selected {value}'

if __name__ == '__main__':
    app.run_server(debug=True)
```

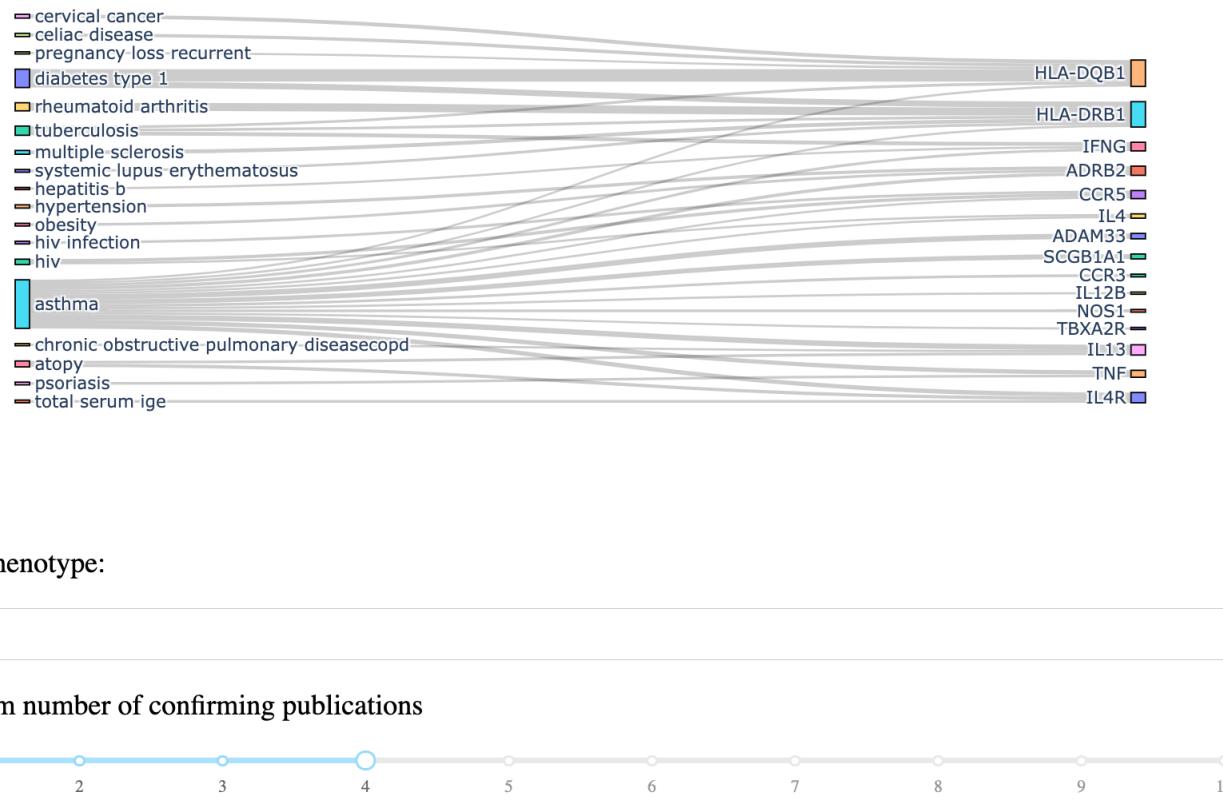
NYC

You have selected NYC



Northeastern University

# Putting it all together.....



An interactive database-driven web application for visualizing gene-disease associations and identifying related phenotypes based on gene overlap.



Northeastern University

# Array Processing with Numpy

---

(DS2500 Review)



Northeastern University

# Types of Arrays

Type	Description	Advantage	Disadvantage	
Lists	Standard built-in python list collection	Simplicity Non-homogeneous	Slower performance Limited functionality	[[1, 2, 3], [4, 5, 6]]
ndarray (NumPy)	High-performance array module for scientific applications.	High performance Flexible operations	Homogeneous values Numeric indexing only	array([1,2,3])
Series (Pandas)	Enhanced one-dimensional array	Custom indexing Missing-value support Many capabilities	Complexity	Wally 87 Eva 100 Sam 94 dtype: int64
DataFrame (Pandas)	Enhanced two-dimensional array (e.g., “tables”)	Flexible support for manipulating tabular data: filtering, sorting	Complexity	A B C 0 10 True 5.4 1 15 False 8.8 2 12 False 6.6



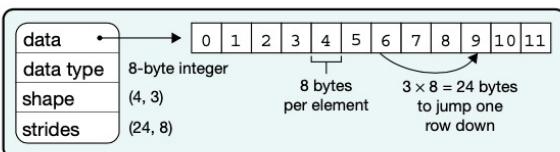
D

# Array operations

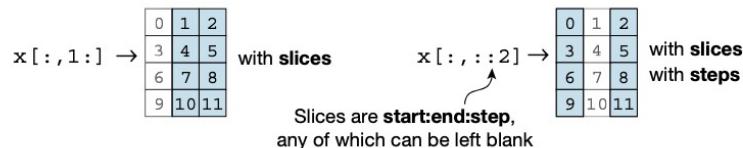
Nature | Vol 585 | 17 September 2020 | 357

## a Data structure

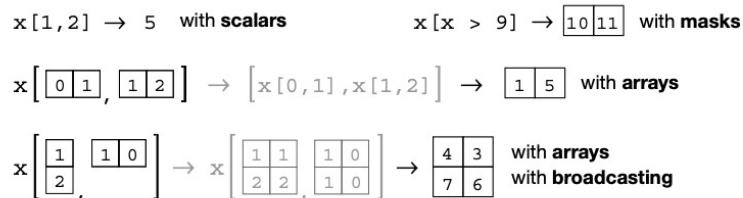
0	1	2
3	4	5
6	7	8
9	10	11



## b Indexing (view)



## c Indexing (copy)



**Fig. 1 | The NumPy array incorporates several fundamental array concepts.**

- a, The NumPy array data structure and its associated metadata fields.
- b, Indexing an array with slices and steps. These operations return a ‘view’ of the original data.
- c, Indexing an array with masks, scalar coordinates or other arrays, so that it returns a ‘copy’ of the original data. In the bottom example, an array is indexed with other arrays; this broadcasts the indexing arguments

## d Vectorization

0	1
3	4
6	7
9	10

1	1
1	1
1	1
1	1

1	2
4	5
7	8
10	11

## g Example

```
In [1]: import numpy as np
In [2]: x = np.arange(12)
In [3]: x = x.reshape(4, 3)
```

## e Broadcasting

0		1	2
3			
6			
9			


0	0
3	6
6	12
9	18

## f Reduction

0	1	2
3	4	5
6	7	8
9	10	11

3
12
21
30

18	22	26
sum axis 0 ↓		
sum axis (0,1) → 66		

```
In [4]: x
Out [4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out [5]: array([4.5, 5.5, 6.5])

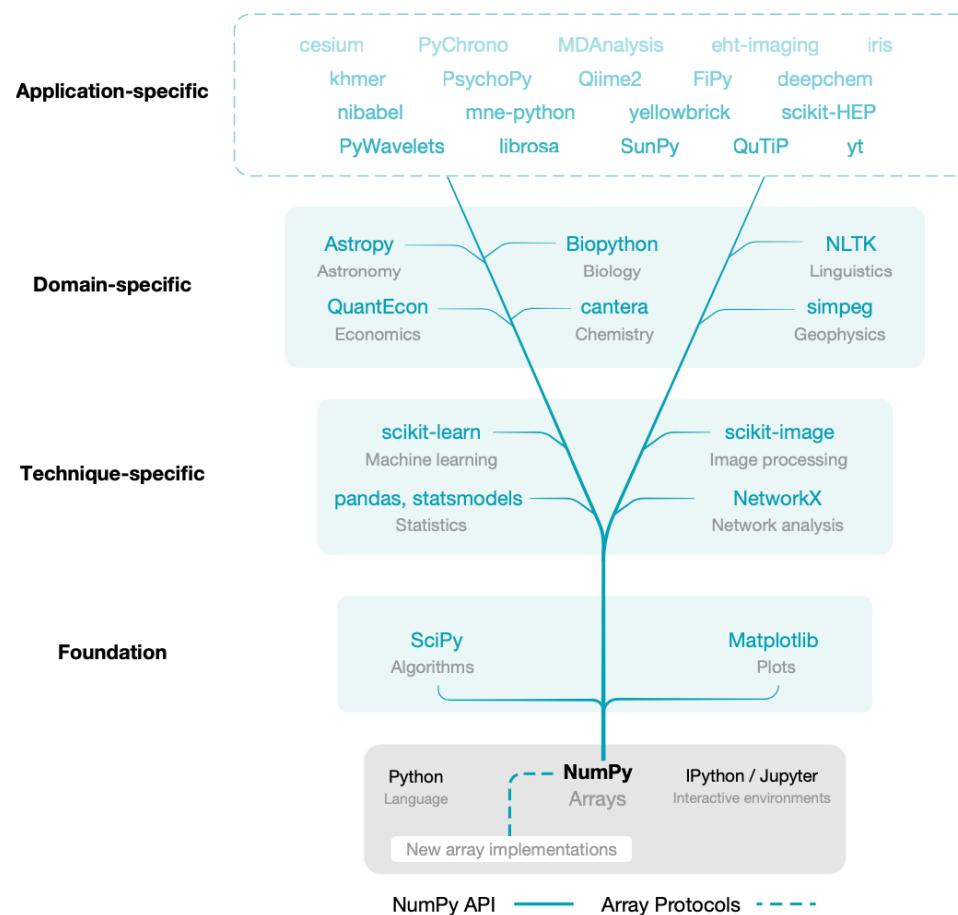
In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out [7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```

before performing the lookup. d, Vectorization efficiently applies operations to groups of elements. e, Broadcasting in the multiplication of two-dimensional arrays. f, Reduction operations act along one or more axes. In this example, an array is summed along select axes to produce a vector, or along two axes consecutively to produce a scalar. g, Example NumPy code, illustrating some of these concepts.



# NumPy is the base of the scientific Python ecosystem



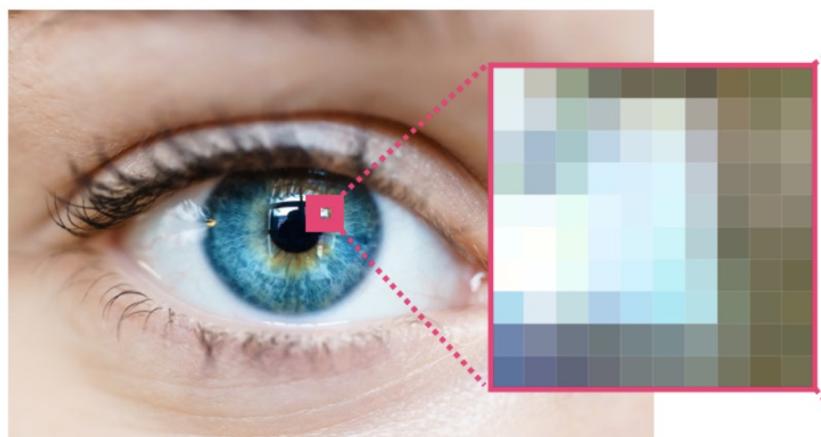
Nature | Vol 585 | 17 September 2020 | 357



**Fig. 2 | NumPy is the base of the scientific Python ecosystem.** Essential libraries and projects that depend on NumPy's API gain access to new array implementations that support NumPy's array protocols (Fig. 3).

# NumPy Image Analysis

If the image is colored, then each pixel is represented by three numbers - a value for each of red, green, and blue. In that case we need a 3rd dimension (because each cell can only contain one number). So a colored image is represented by an ndarray of dimensions: (height x width x 3).



233	188	137	96	90	95	63	73	73	82	
237	202	159	120	105	110	88	107	112	121	109
226	191	147	110	101	112	98	123	110	119	142
221	191	176	182	203	214	169	144	133	145	155
185	160	161	184	205	223	186	137	147	161	140
181	174	189	207	206	215	194	136	142	151	133
246	237	237	231	208	206	192	122	143	144	111
254	254	241	224	199	192	181	99	122	117	107
239	248	232	207	187	182	184	110	114	110	113
193	215	193	167	158	164	181	114	112	111	105
113	119	110	111	113	123	135	120	108	106	82
93	97	91	103	107	111	122	112	104	114	113



# Animation

---

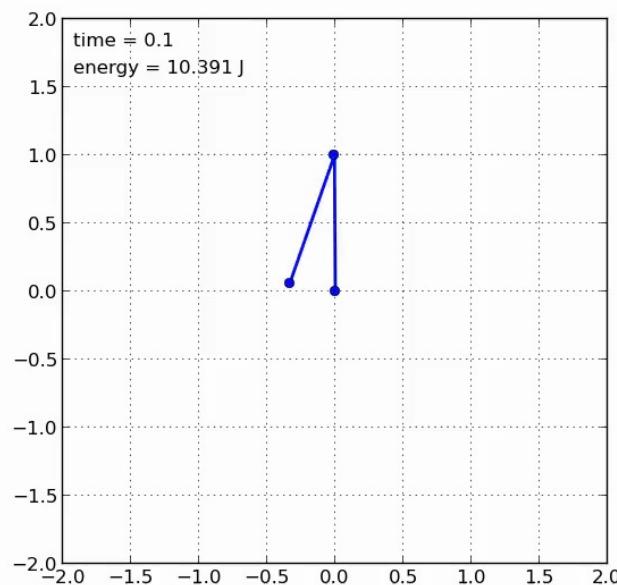
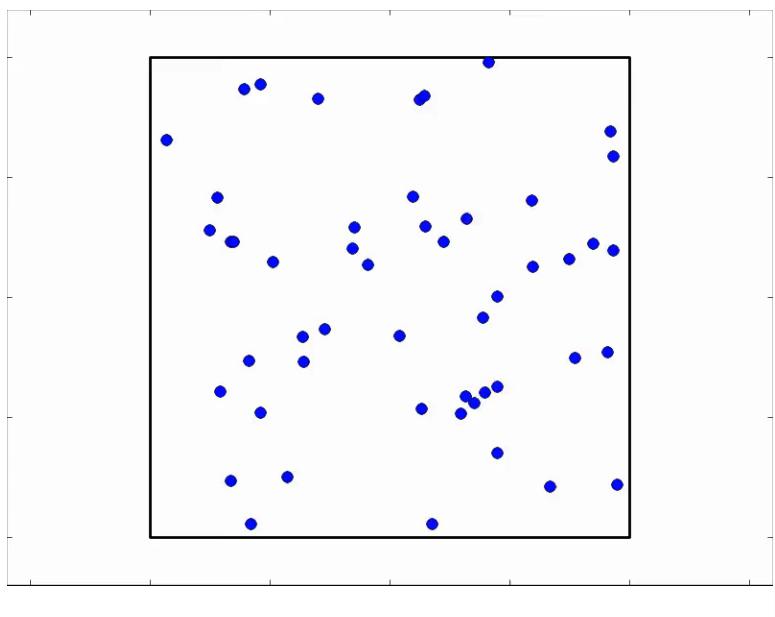
Created animated visuals using Matplotlib



Northeastern University

# Example Animations

---



Source: <https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/>



Northeastern University

# FuncAnimation

---

## matplotlib.animation.FuncAnimation

```
class matplotlib.animation.FuncAnimation(fig, func, frames=None, init_func=None,
fargs=None, save_count=None, *, cache_frame_data=True, **kwargs)           [source]
```

Makes an animation by repeatedly calling a function *func*.

 Note

You must store the created Animation in a variable that lives as long as the animation should run. Otherwise, the Animation object will be garbage-collected and the animation stops.

**fig** : plt.figure()

**func** : The animation function – called each frame

**frames** : The number of frames

**fargs** : Additional arguments to *func* (if needed)

**interval** : Time (ms) between frames



# Artificial Life

---

Life as it could be



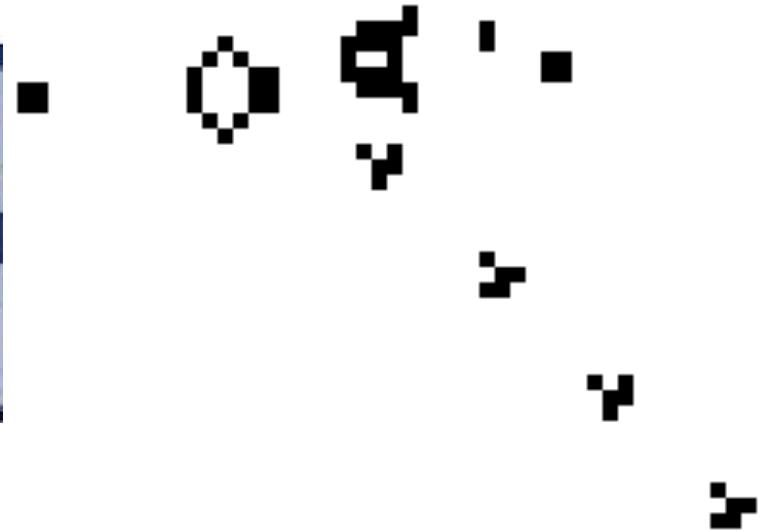
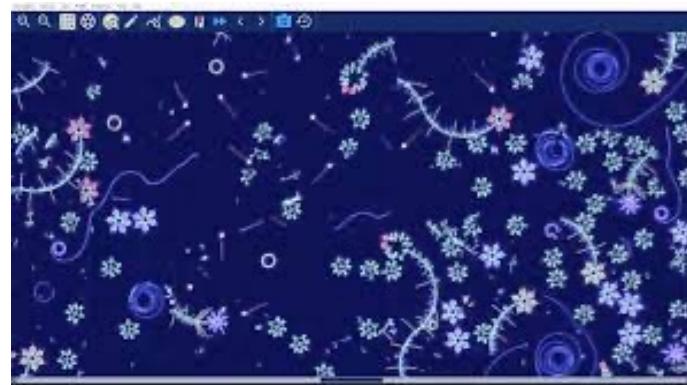
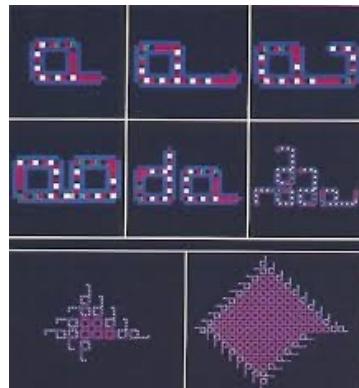
Northeastern University

# Artificial Life Defined

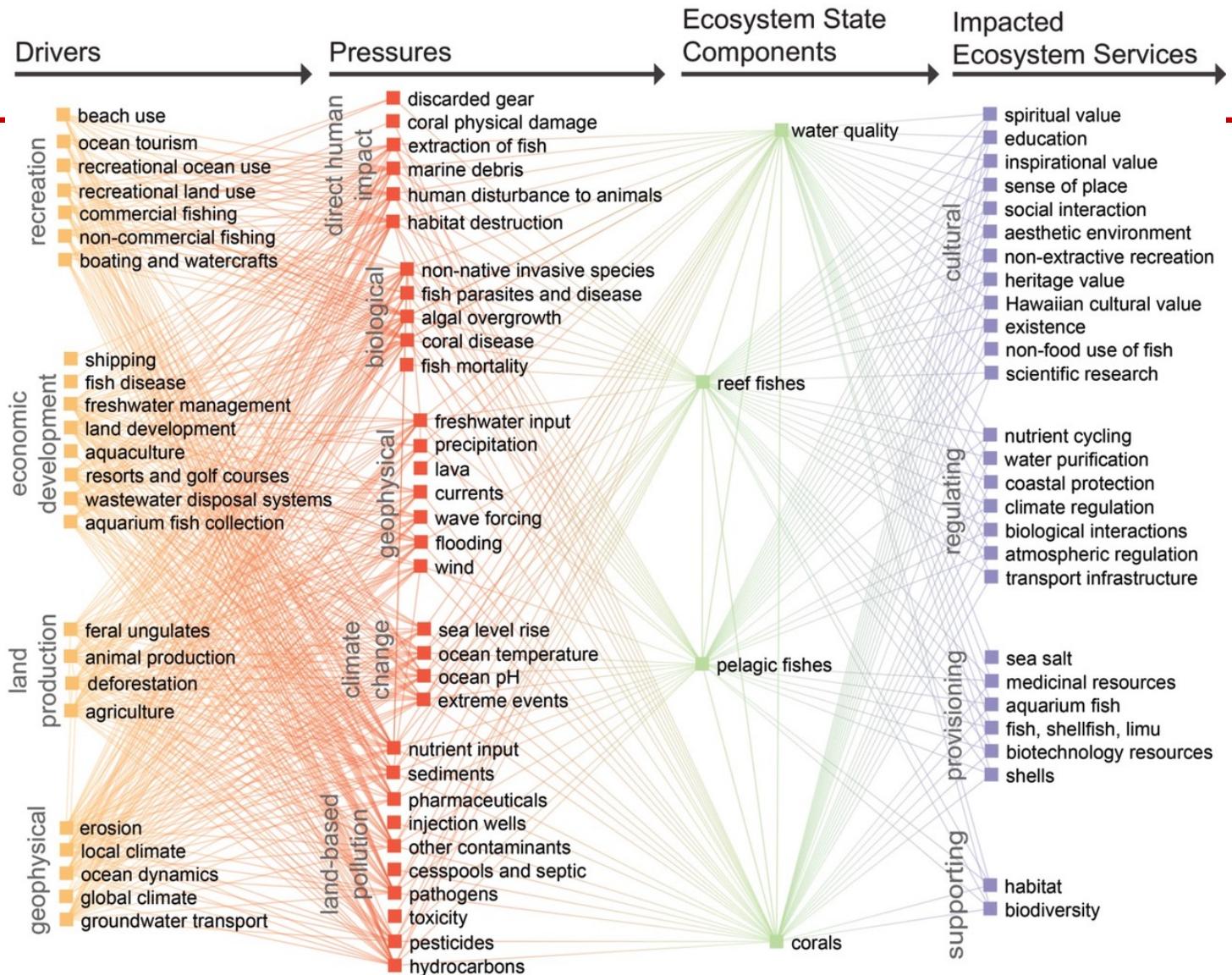
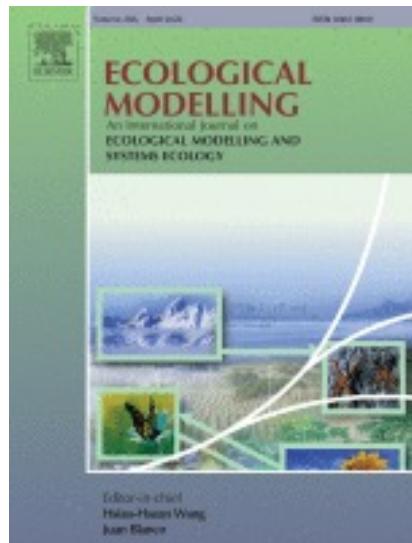
---

**Artificial life** (often abbreviated **ALife** or **A-Life**) is a field of study wherein researchers examine systems related to natural life, its processes, and its evolution, through the use of simulations with computer models, robotics, and biochemistry.

The discipline was named by [Christopher Langton](#), an American theoretical biologist, in 1986.



# Ecosystems



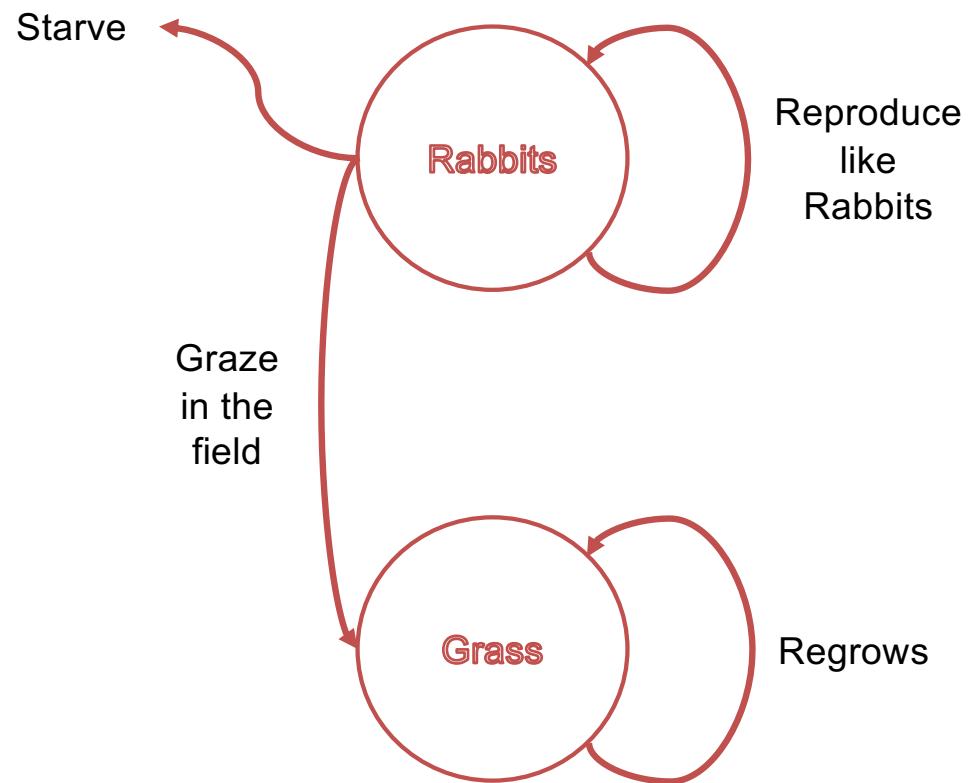
Northeastern University

# Conceptual Models

A **conceptual model** identifies the qualitative (rather than quantitative) relationships among the various entities, suggesting causal connections without precisely defining the nature of the relationships in precise mathematical terms.

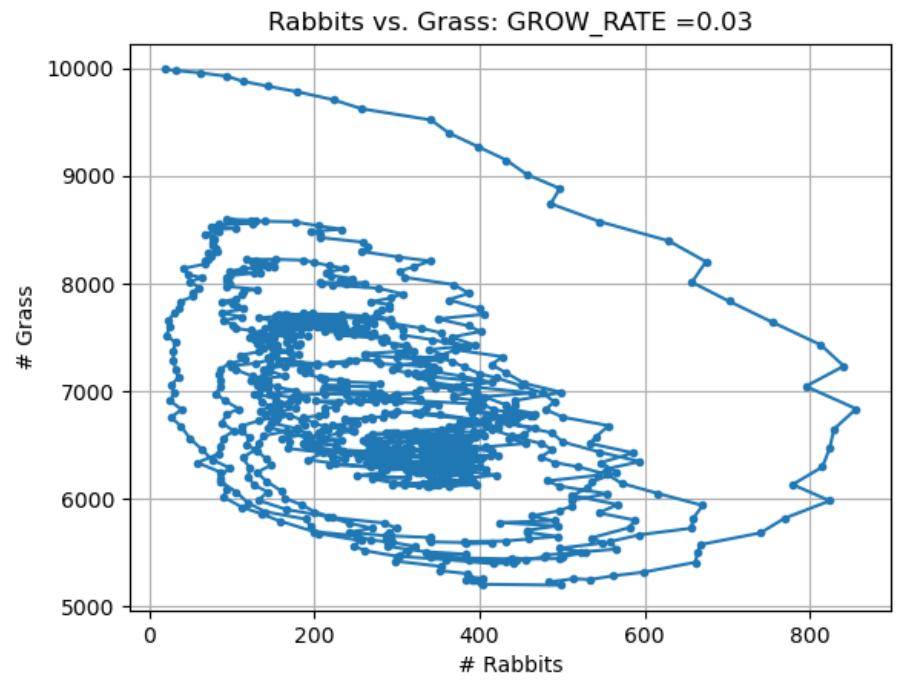
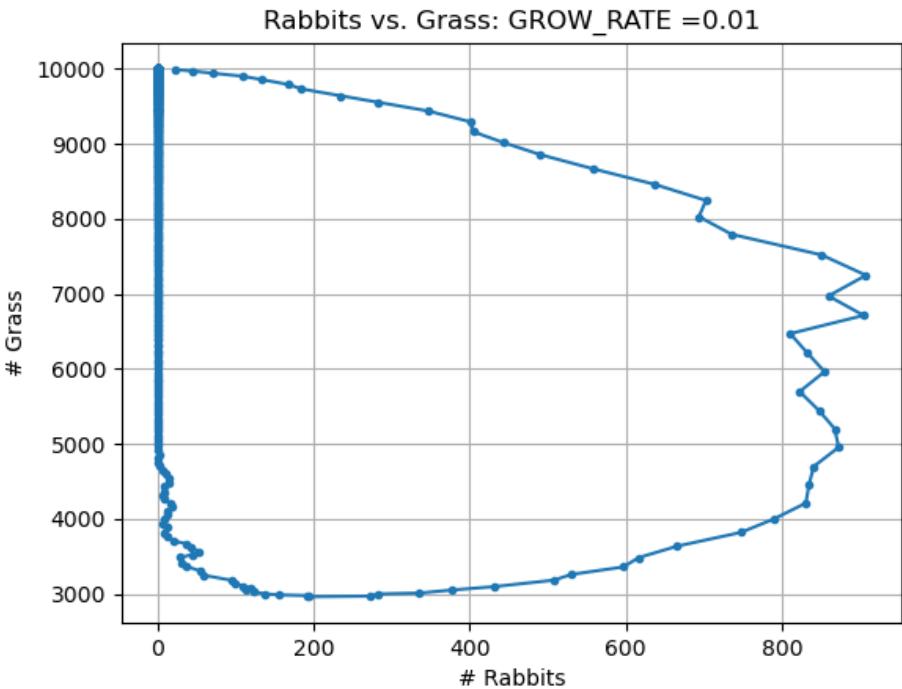
The conceptual model may serve as a starting point for understanding the underlying dynamics of **complex systems**.

From the study and simulation of conceptual models, more quantitative mathematical models may emerge. Until then, **simulation** may be used to validate the underlying assumptions of the conceptual model or to explore possible **emergent phenomena** or the **stability** of the underlying system.



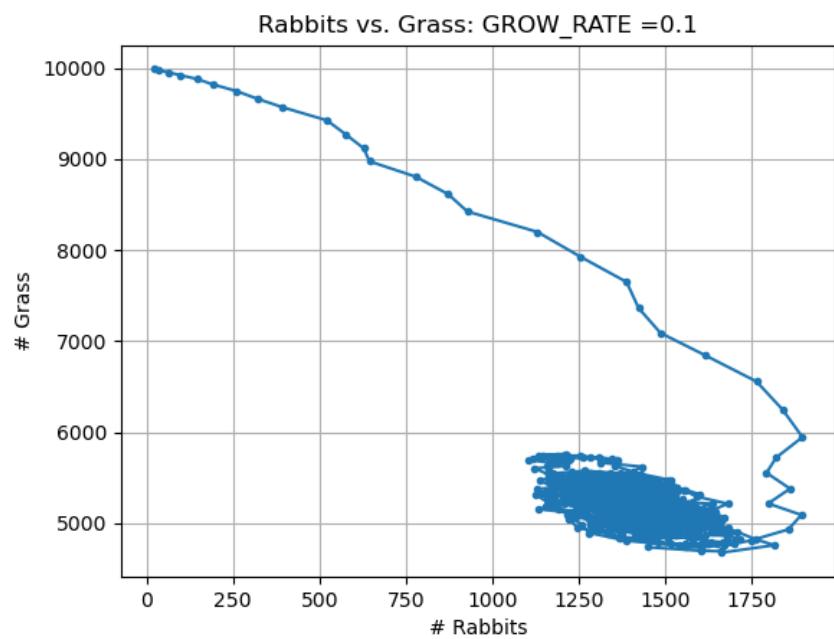
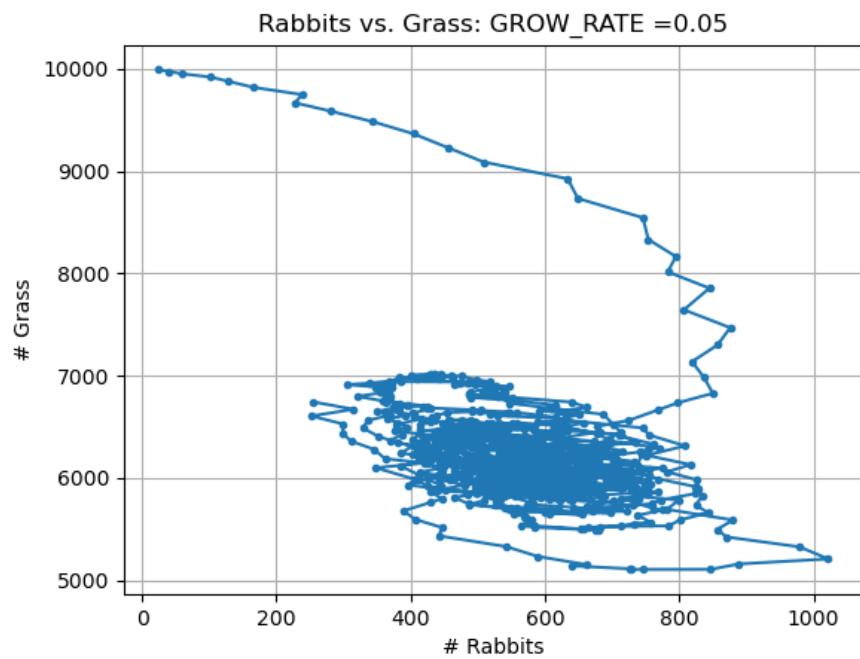
# Rabbits vs. Grass

---



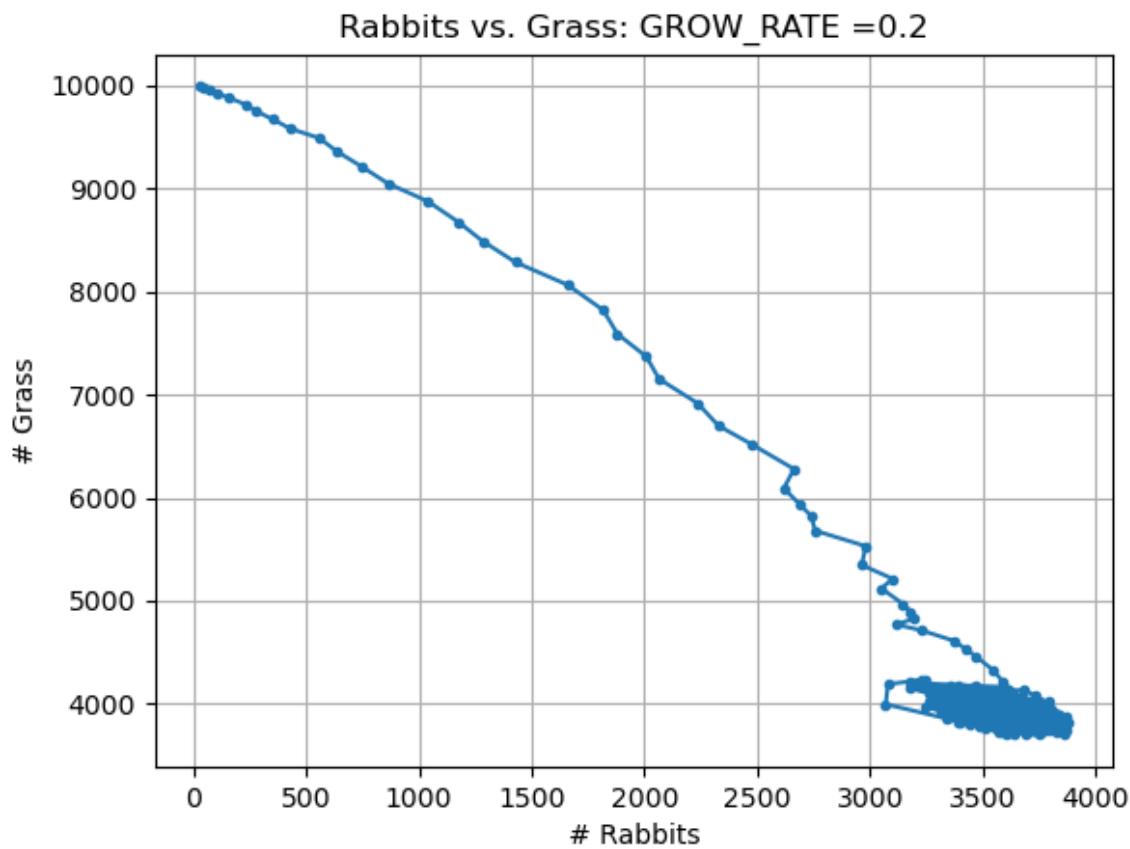
# Rabbits vs. Grass

---

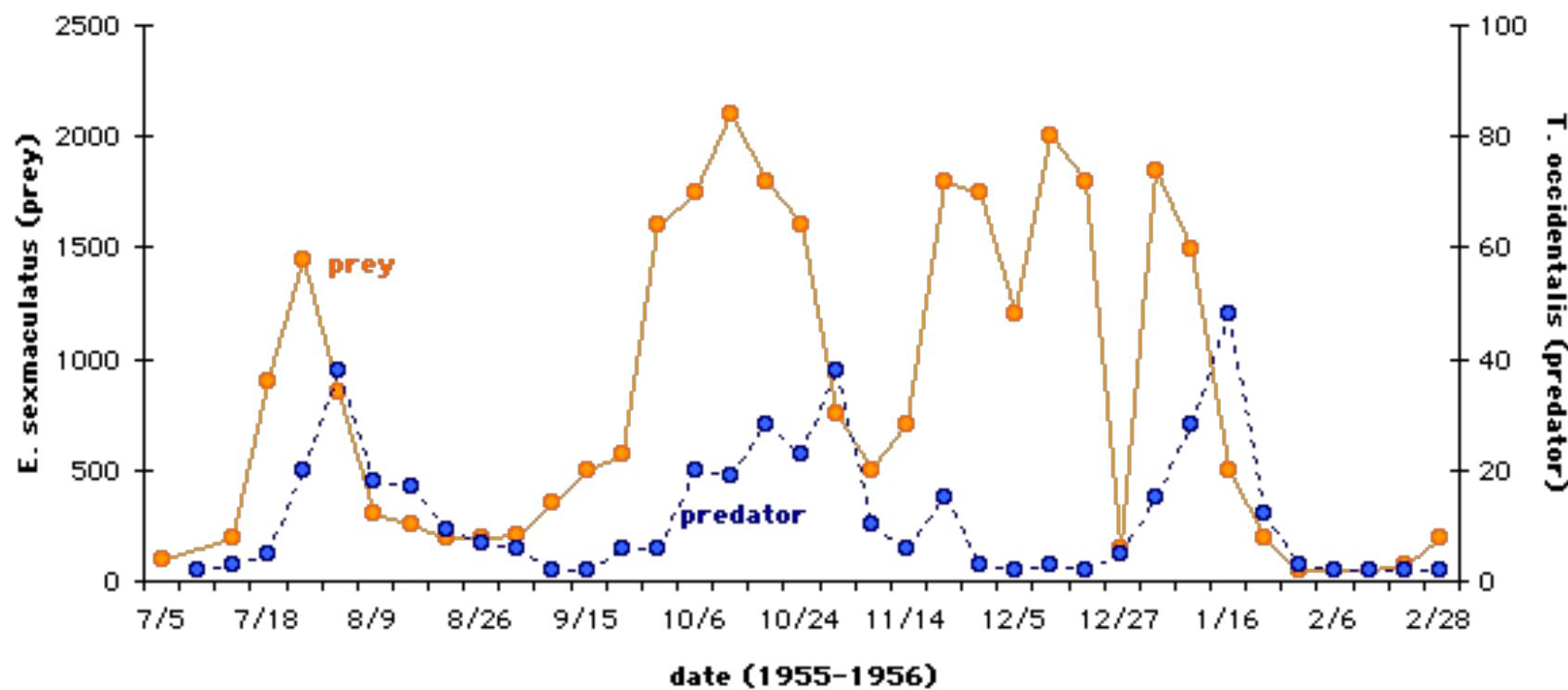


# Rabbits vs. Grass

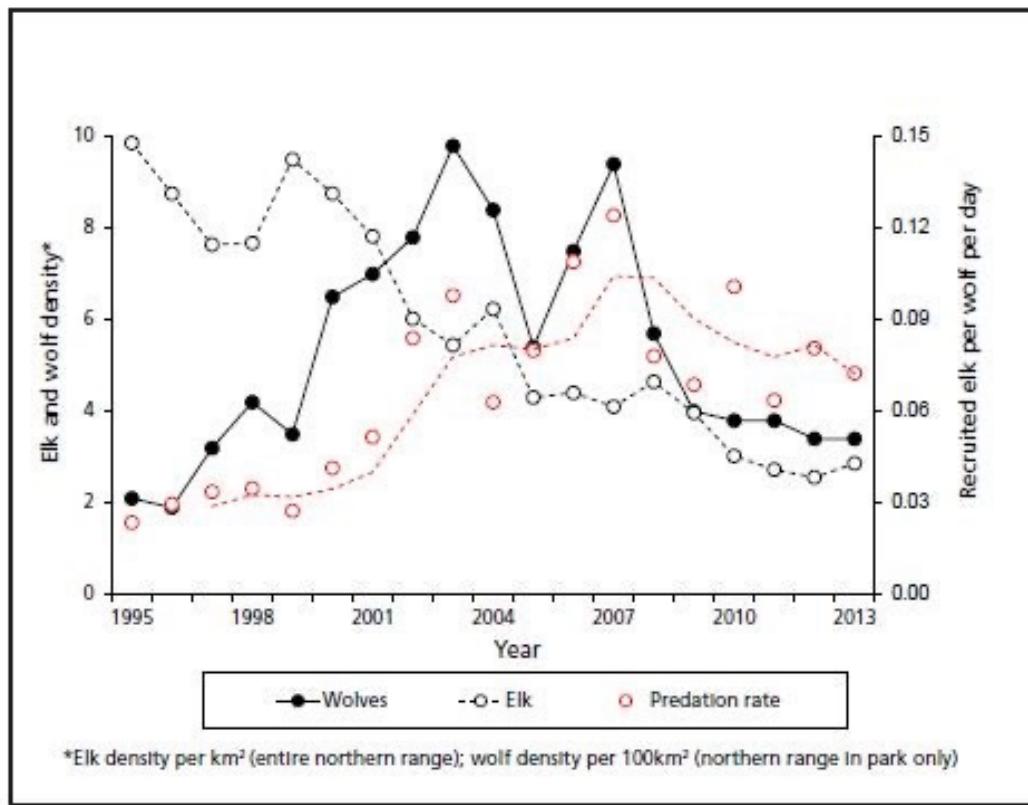
---



# Mite vs. Mite



# Wolves vs. Elk in Yellowstone National Park



# Lotka-Volterra Equations

---

where

$$\begin{aligned}\frac{dx}{dt} &= \alpha x - \beta xy, \\ \frac{dy}{dt} &= \delta xy - \gamma y,\end{aligned}$$

- $x$  is the number of prey (for example, rabbits);
- $y$  is the number of some predator (for example, foxes);
- $\frac{dy}{dt}$  and  $\frac{dx}{dt}$  represent the instantaneous growth rates of the two populations;
- $t$  represents time;
- $\alpha, \beta, \gamma, \delta$  are positive real parameters describing the interaction of the two species.

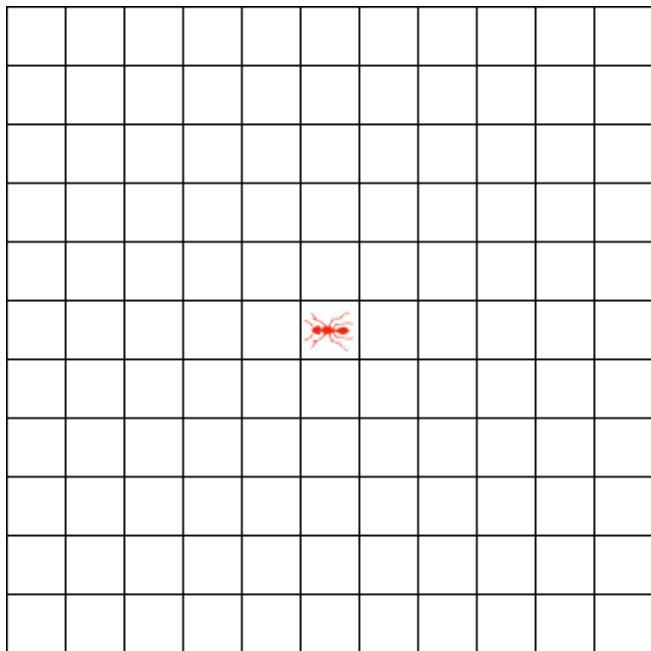


# Langton's Ants

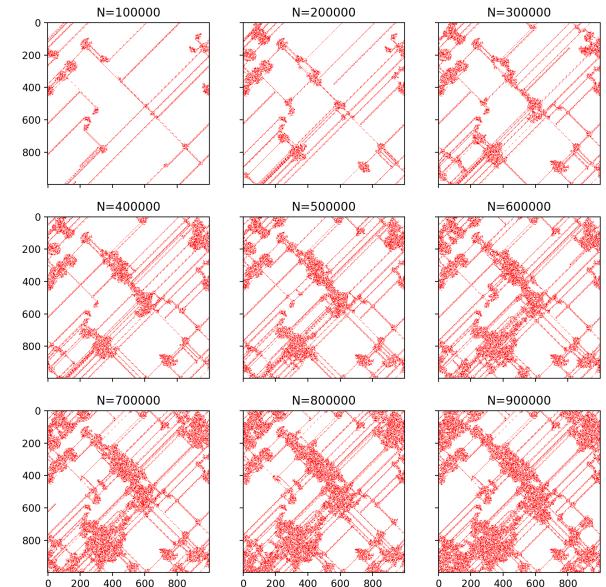
Modeling ants responding to pheromone signals.

An example of how the repeated application of simple rules → complexity

**White:** Turn Right, **Red:** Turn Left, **White**, 1 Step



Langton's ant after 11,000 steps. A red pixel shows the ant's location.



# Simulation and Modeling with Discrete Random Variables

---



Northeastern University

J. Rachlin  
Northeastern University

# Please build our town a bridge!

---



The cost is **\$25 million.**  
Guaranteed!



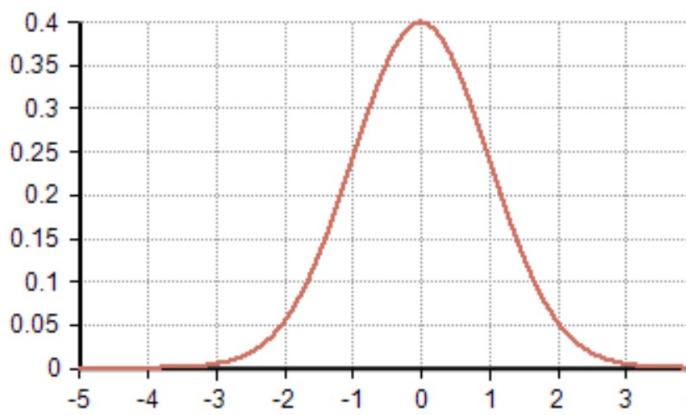
Northeastern University

# Please build our town a bridge!

---



The expected cost is **\$20 million**.  
I'm 95% sure it will cost  
between \$10-\$30 million.



# Please build our town a bridge!



The *expected cost* is just **\$15 million**.  
and most likely we can do it for **\$5 million!**

***Translation: If we built you 1000 bridges the average cost would be \$15 million.***

$p=0.95$ : cost = **5 million**

$p=0.05$ : cost = **205 million <<ouch!>>**

$$\text{Avg Cost} = (0.95)(5) + (0.05)(205) = 15$$



# Auto Insurance

---

Plan	Premium	Deductible	Expected Cost $P(\text{accident})=0.05$
A	\$100	\$500	\$1225
B	\$80	\$1000	\$1010

$$\text{Plan B: } 0.95(12 \times \$80) + 0.05(12 \times \$80 + \$1000) = \$1010$$

Is Plan B necessarily the better plan for you?  
Can you afford the larger deductible?  
**What is your tolerance for *risk*?**



# Risk and Uncertainty in Business Decision-Making

---

Should we acquire this company?

Should we hire this candidate and if so, at what salary?

Should I accept this investment offer for 5% of my company?

Should we move our data center to the cloud?

Can we upgrade our manufacturing facility?

How much should we spend on advertising?

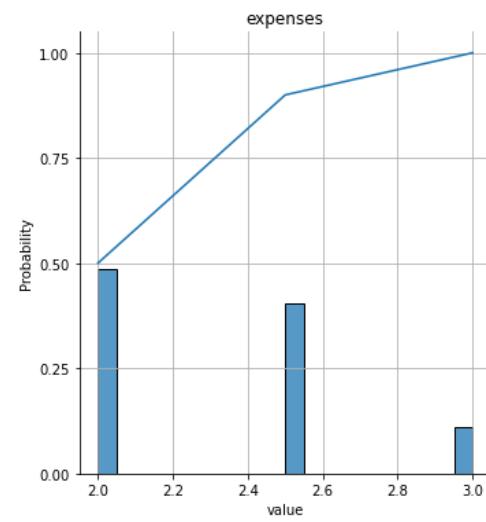
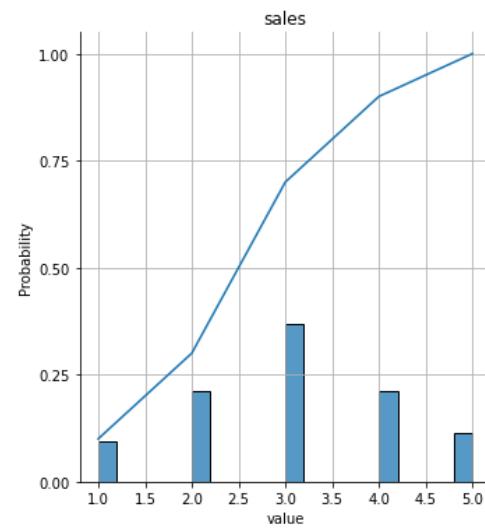
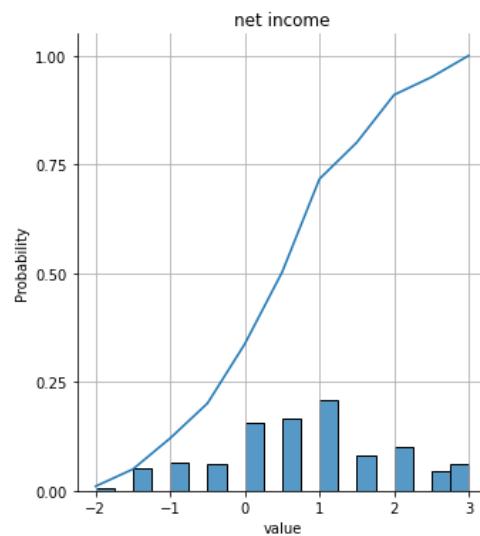
Should we focus on profits or growth?

What products should we stock?



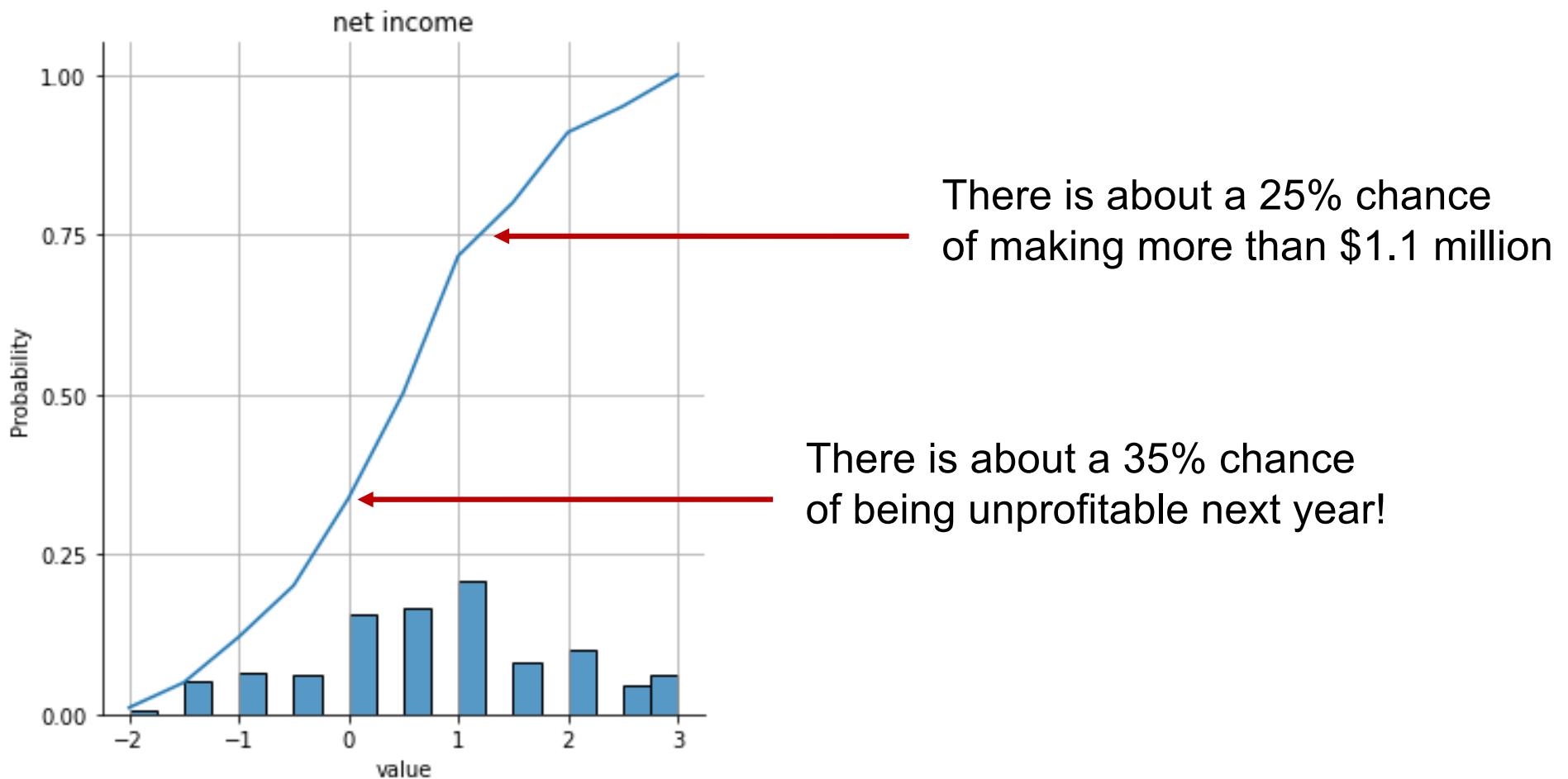
# Business Forecasting: Net Income

$$\begin{array}{rcl} \textbf{Net Income} & = & \textbf{Sales} - \textbf{Expenses} \\ 0.7 \text{ Billion} & = & 3.0 \text{ Billion} - 2.3 \text{ Billion} \end{array}$$



# Business Forecasting: it's complicated

---



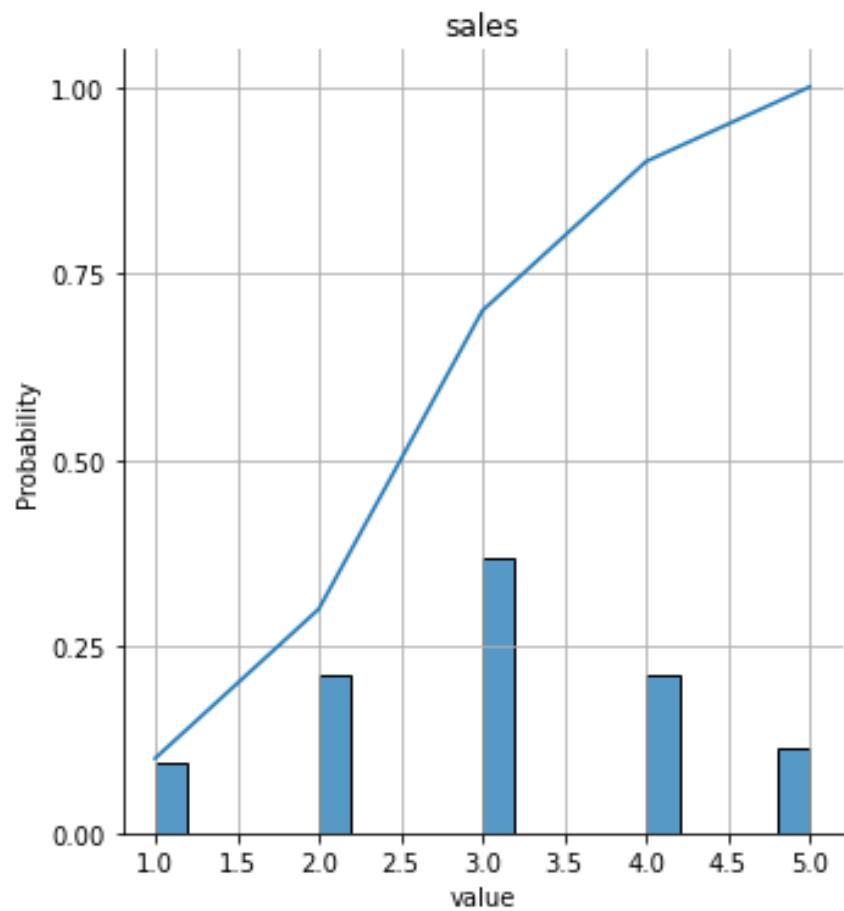
# Random Variables

---

**sales =**

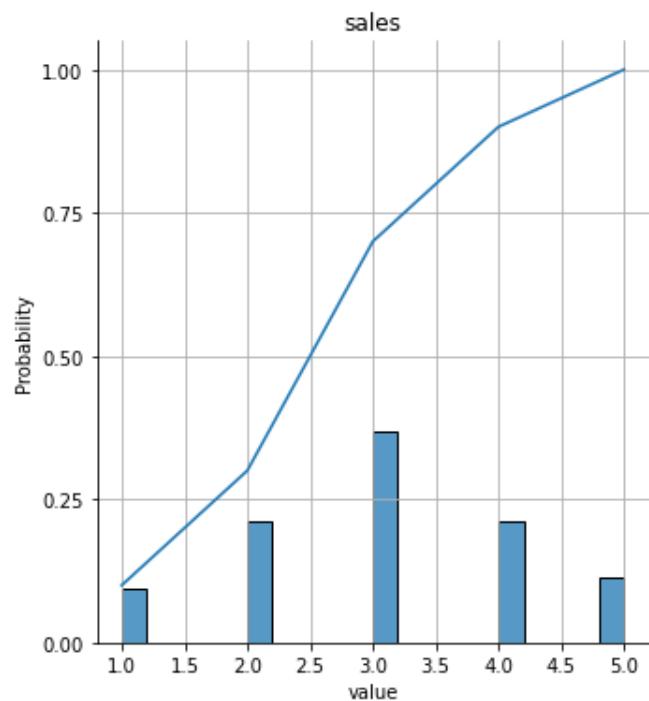
Sales is not a single number.  
It's a *probability distribution!*

It takes on different values,  
each with a specified  
probability.



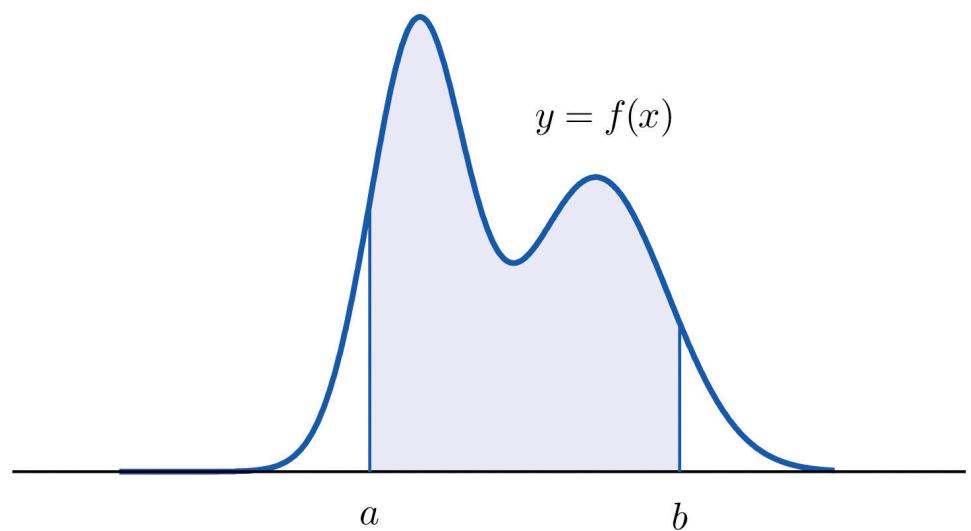
# Discrete and Continuous Random Variables

---



Discrete ✓

$$P(a < X < b) = \text{area of shaded region}$$



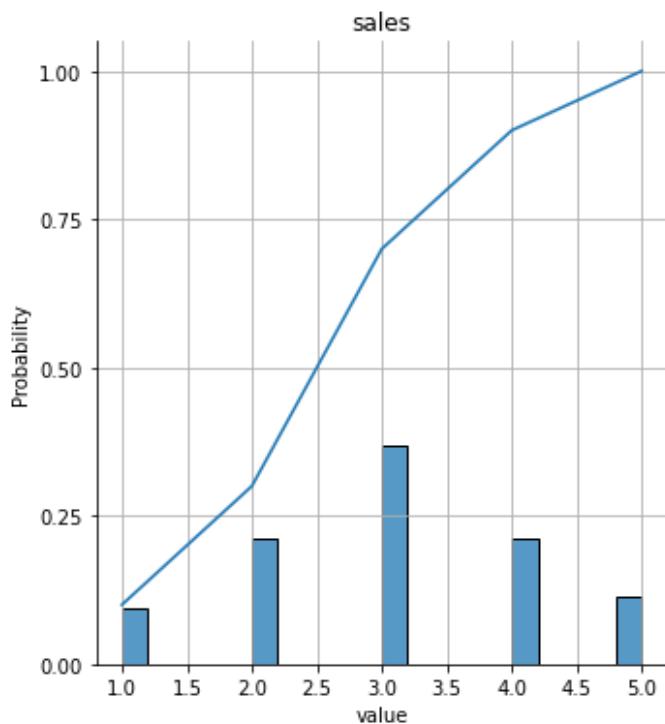
Continuous



Northeastern University

# Discrete Random Variables are objects!

---



The distribution stored inside the object is simply a dictionary where:

Keys are **outcomes**

Values are **probabilities** that add to 1.0

```
{  
    1 : .10, 2 : .20, 3 : .40,  
    4 : .20, 5 : .10  
}
```



# Mathematical expressions are now *simulations*!

---

**sales - expenses = net income**

$$5 - 2 = 3$$

$$4 - 2 = 2$$

$$3 - 2.5 = 0.5$$

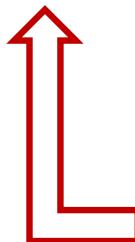
$$3 - 2 = 1$$

$$2 - 3 = -1$$

.

.

etc.



A new discrete  
random variable!



# But a simulation isn't necessary if ...

---

- our random variables are *discrete*
- they represent *independent* events:  $p(\mathbf{X} \text{ and } \mathbf{Y}) = p(\mathbf{X}) p(\mathbf{Y})$

		Y		
		0 (.20)	2 (.50)	5 (.30)
		1 (.60)	3 (.30)	6 (.18)
X	1 (.60)	1 (.12)	3 (.30)	6 (.18)
	3 (.40)	3 (.08)	5 (.20)	8 (.12)

$Z = X + Y$

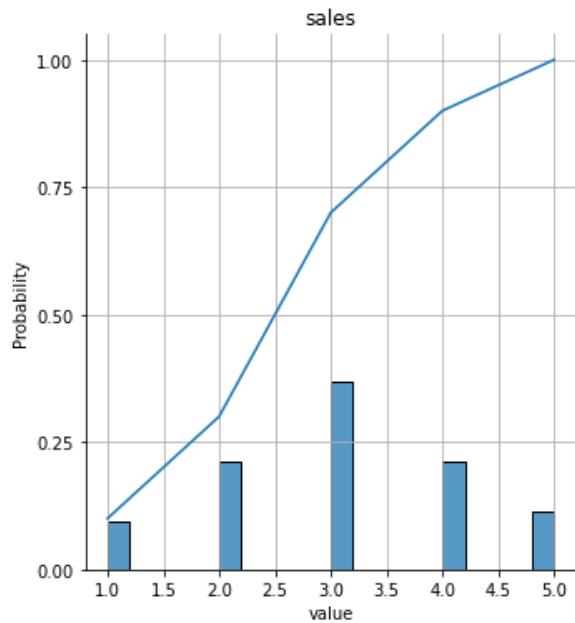


# Expected value of X, denoted as E[X].

---

The expected value is basically a weighted average.  
We are weighting each possible value by its probability.

$$E[\text{sales}] = 1(0.1) + 2(0.2) + 3(0.4) + 4(0.2) + 5(0.1) = 3.0$$



$$E[X] = \sum x_i p(x_i)$$

$x_i$  = The values that X takes

$p(x_i)$  = The probability that X takes the value  $x_i$